UHASSELT
KNOWLEDGE IN ACTION
www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Predicting Prices for Fastening Parts: An Analysis of Bolts, Nuts, Screws , Link and Contracts Factors

Entzi Rakipi

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Data Science

SUPERVISOR :
Prof. dr. Dirk VALKENBORG

SUPERVISOR :
Stijn MOONS

2024
2025

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Predicting Prices for Fastening Parts: An Analysis of Bolts, Nuts, Screws , Link and Contracts Factors

Entzi Rakipi

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Data Science

SUPERVISOR :

Prof. dr. Dirk VALKENBORG

SUPERVISOR :

Stijn MOONS

Faculty of Sciences School for Information Technology

Master Thesis

Cost Prediction of Automotive Fastening Parts

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Data Science

Authors:
Entzi Rakipi

Supervisor:
Dirk Valkenborg

Master of Statistics and Data Science

2023-2024

22nd January 2025

Contents

1 Acknowledgment

I would like to start by appreciating Prof. Dirk Valkenborg's assistance, who was essential to the success of this project and not only. I am grateful for the opportunity of working on this endeavor with his outstanding supervision. He helped me expand my understanding and enthusiasm in the subject, while working on my thesis. He was always available. Answering my queries and giving constructive comments. I was and am so happy that I had the chance to have you as my supervisor.

Additionally, is worth mentioning all the effortless work of the professors, that I have met through this Master degree and beside them, managed to expand the horizons of my knowledge and become who I am today. There is no doubt, that they are my role models, for what I want to pursue in the future.

Externally, I want to show my gratitude to my co-workers, with whom I had the pleasure to collaborate for this project and I appreciate the trust they showed me throughout this project and of course the last 2 years, where I had to combine my studies with work. Last but not least, I will never forget the friendships that I have created. The happy and not so happy moments, that we shared together through this journey. This Master had no borders. No borders had the understanding, the emotional support and the compassion we gave to each other, no matter country, culture or background.

Thankful to each one of you! I would not change anything or anyone!

2 Abstract

This thesis explores cost prediction for automotive fastening parts, such as screws, bolts, and nuts, to support strategic decision-making for Facil.

The project begins by integrating diverse data sources, including technical specifications, supplier information, and external factors like oil prices, to create a detailed dataset.

Data cleaning and analysis were critical steps, addressing challenges such as missing values, outliers, and inconsistencies. Using this refined data, three machine learning models—K-Nearest Neighbors (KNN), Random Forest, and Gradient Boosting—were applied to predict costs, with their performance evaluated using Root Mean Squared Error (RMSE). Among these, the Random Forest model showed the best predictive accuracy, particularly for screws and nuts. For bolts, some variability in the data posed additional challenges.

This thesis highlights the importance of rigorous preprocessing, thoughtful feature selection, and clear evaluation metrics in building reliable predictive models. The study concludes with practical recommendations for improving data management and integrating these predictive approaches into real-world applications.

Key Words: Automotive Part Number, Cost Prediction, Predictive Modeling, Data Visualization, Random Forest, KNN, Gradient Boosting, Hyper-Parameter Optimization

3 Introduction

In today's data-driven world, advancements in statistics and technology have made it increasingly possible to leverage machine learning for forecasting. Forecasting, a branch of predictive analytics, is widely applied in fields like meteorology, polling, and business, where it provides valuable insights into current trends and potential future outcomes. While predictive models cannot provide all the answers on their own, combining them with domain-specific knowledge—such as business acumen—allows organizations to make more informed strategic decisions. Forecasting is particularly useful in businesses for creating strategic plans, managing finances, identifying risks, allocating resources efficiently, and understanding the effects of policy changes or new initiatives on overall performance. [**1**]

This thesis focuses on cost forecasting for Facil, a company at the heart of automotive fastening solutions. Established in 1999, Facil is a strategic partnership between Kamax, a specialist in bolts and screws, and A. Raymond, an expert in engineered metal and plastic fasteners. Facil serves as a key intermediary between automobile manufacturers and fastener suppliers, offering a comprehensive range of services to streamline the development and production of automotive vehicles. These services include:[**2**]

1) **Engineering service**: support for design, development, and production implementation of fasteners for automotive manufacturing.

2) **Purchase service**: procurement of a diverse range of fasteners through a network of reliable suppliers, ensuring quality and availability.

3) **Quality service**: inspection and testing of fasteners to match or exceed benchmarks set by manufacturers and suppliers.

4) **Supply Chain service**: ensuring seamless delivery and stock monitoring to prevent production delays.

5) **E-Commerce service**: connecting fastener suppliers with automobile manufacturers for streamlined procurement and collaboration.

Through these services, Facil optimizes the procurement and supply of fasteners used in vehicle manufacturing, making it a vital player in the automotive supply chain.

3.1 Objective

The goal of this thesis is to develop predictive models that forecast the costs of fastening parts such as screws, bolts, and nuts, providing Facil with data-driven insights to enhance decision-making and operational efficiency.

3.2 Flowchart

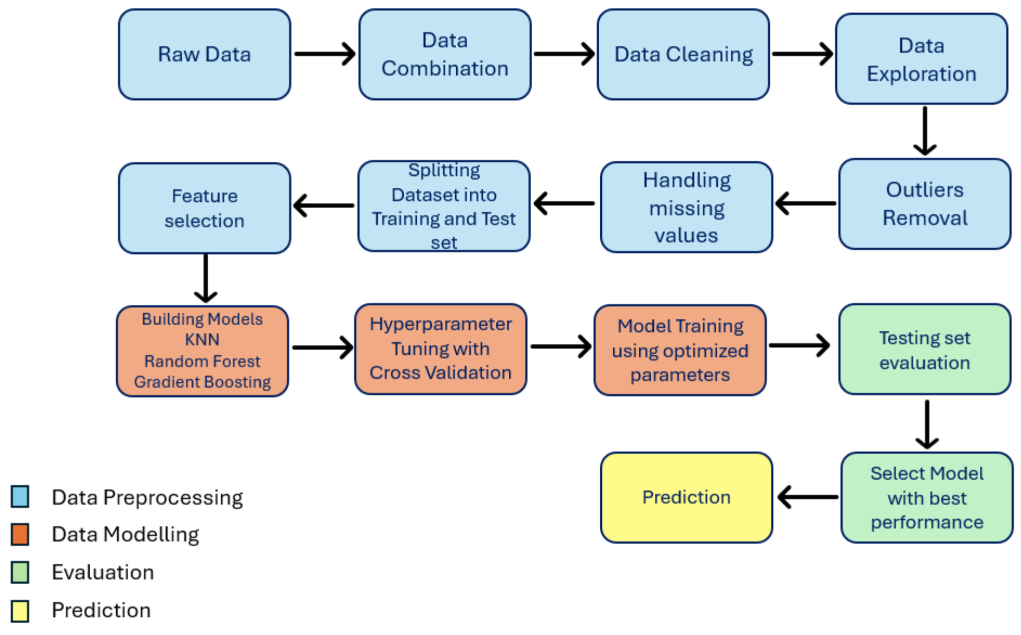
Raw Data
Data Combination
Data Cleaning
Data Exploration
Outliers Removal
Handling missing values
Splitting Dataset into Training and Test set
Feature selection
Building Models KNN Random Forest Gradient Boosting
Hyperparameter Tuning with Cross Validation
Model Training using optimized parameters
Testing set evaluation
Select Model with best performance
Prediction
Data Preprocessing
Data Modelling
Evaluation
Prediction

Figure 1: Steps of the process

To predict the costs of fastening parts, such as bolts, nuts, and screws, a series of structured steps were implemented. First, raw data was gathered from various departments within the organization. The required datasets were then combined to form a comprehensive final dataset that would serve as the foundation for the cost prediction models. Since the data was significantly polluted, data cleaning was carried out to remove inconsistencies and errors.
Next, data exploration was performed to allow the company to better understand the underlying patterns and structures in the data. During this process, multiple outliers were identified and removed to improve the accuracy of the prediction models. Missing values in the datasets were also addressed using appropriate handling techniques to ensure the completeness and reliability of the data.
After the data was preprocessed, it was split into training and testing sets. This division ensured

that the model could be trained on one subset of the data and evaluated on unseen samples, guaranteeing its ability to generalize to new situations. Feature selection followed, focusing on identifying the top 10 most impactful features within each of the three datasets. This step allowed the models to prioritize the variables most strongly influencing the cost predictions.

With the preprocessed and refined dataset, three machine learning models—K-Nearest Neighbors (KNN), Random Forest, and Gradient Boosting—were developed. These models underwent hyperparameter tuning using cross-validation techniques to optimize their performance. The training phase involved utilizing these optimized parameters to build models that achieved the best possible results.

Once the models were trained, their performance was evaluated on a separate test dataset to assess their predictive accuracy. Root Mean Squared Error (RMSE) was used as the key performance metric, and the model delivering the best results was selected as the final predictive tool. The chosen model was then applied to make predictions, with its outcomes visualized in a graph illustrating the relationship between predicted prices and actual prices.

4 Data Management

This section will examine, in detail from the outset, including how the data was gathered, which data specifically was obtained, and which department provided the data. Every step taken—including comprehension, evaluation, and action—will be discussed, with instances used to illustrate each point. When we reach the final three matrices, the combination and comments have also been documented. Finally, data visualization is displayed for various combinations.

4.1 Data Collection

The data has been provided from the Facil company, combined the updated information from different departments. In detail:
- the base idea has been provided from the Strategic department.
- the technical information has been provided from the Engineering department.
- as for Raw Material information has been provided from the Purchase department - the updated information regarding the prices of the fastening parts has been provided from the IT department.
- last but not least, regarding the external factor, oil index based on the Federal Reserve Bank [**4**].

4.2 Data Cleaning and Transformation

To be able to come closer to the target of creating a final metrics, on which we clearly have in disposal all the variables that seem interesting for the aim of cost prediction, different excels were taken into account.
Before moving to the combination of the excels, first step after having a good understanding of the data, was to clear the data.
1) Starting with the excel file called, ***facfasxppsparts***, which contains numerical and categorical data (Categorical: DISPLAYNAME, XPPSPARTNR, CLIENTCOATING, FACILFINISH, CATEGORY and Numerical: FACILPARTID, COATINGID) and in total had an amount of 24012 records, the following steps were taken:
Removed the duplicates such as the following: PN E802263S72M and PN E802263S72MK are exact same Part Numbers, at the end we are going to choose PN E802263S72MK since this is

DISNAME	XPPSPN	FPID	CLIENTCOATING	FFINISH	COATINGID	CATEGORY
XPPS	E802263S72M	010012	S72M	Mech - 96	193	01.Bolts
XPPS	E802263S72MK	010012	S72M	Mech - 96	193	01.Bolts

Table 1: Example table from facfasxppsparts

the valid PN to base on in the system for plant 000(Genk).
On column A: DISNAME, there were shown multiple options such as CNHi, Daimler China, Fisker, Ford Otosan, SAP, XPPS and etc. but the only one that is really worth focusing on is the XPPS as it corresponds to the information downloaded from the main program that is called InforXpert. Hence, the rest were deleted. On column H: Category focused on only in the three domain categories, which were Screws, Nuts and Bolts. The rest of the the categories were deleted as well. Hence the final columnns that were kept are the following:

XPPSPN	FPID	CATEGORY
972083	030188	03.Screws
E802263S72MK	010012	01.Bolts
STD99004207	070034	07.Nuts

Table 2: Three main columns of interest (facfasxppsparts)

For more details, see **Table 17**.

2) Another excel which ended up to be the domain one is, ***copy of pc eu no name***, which contains categorical, numerical data and time. For more details regarding the data see **Table 04**.

The following steps were counted:

The scratch PNs which were distinguished from their price were deleted and additionally, the Proto PNs were deleted as well. The ones that were easy to distinguish, were the ones that started or ended, either with PR or PM. Additionally, focused only in the main 9 European plants, which are:

000:Genk, **001:Valencia**, **002:Saarlouis**, **004:Bourg en Bresse**, **005:Tuve**, **006:Blainville**, **007:Umea**, **008:Gent Trucks**, **009:Gent Part**. Regarding plant 003: Munich, which is an old plant and for that reason after the clearance is kept almost none data.

FIRM	WKNR	TENR	LINR	GUVO	GUBI	LIMG	BEPR	WACD
1	000	W700069S442K	10031	20071001	20081130	8150	31,7	EUR
1	009	1501896	20816	20130501	20210731	17803	13300	SEK
1	009	1501896	20816	20210801	99999999	11200	18925	SEK

Table 3: (copy of pc eu no name)

COLUMN TITLE	DESCRIPTION
FIRM	Likely represents the company code (Europe)
WKNR	Represents the plant or warehouse in Europe
TENR	Indicates the part number
LINR	Supplier number.
GUVO	Represents the start date from which a particular entry, is active or applicable
GUBI	Represents the end date until which the entry (e.g., part validity) is valid
LIMG	Indicates the quantity of items supplied in a batch/order
BEPR	Shows the purchasing price or cost per unit of the item
WACD	Represents the currency of the purchasing price (e.g., EUR, GBP)

Table 4: Description of excel copy of pc eu no name

Still even in this excel, there were duplicates which were cleared.

As well, the purpose was to focus only on latest available price of each part number, which indicates the latest active purchase contract.

Additionally, took only into account, given the rows with the latest prices, kept only the combinations of each unique part number with each unique supplier associated, in each plant. That

way each line is a special combination and any other duplicate has been removed.
In addition, removed column ’FIRM’, because as mentioned all rows are one.The most important column is ’WKNR’ as it shows all plants from 000 till 009 are all European plants, removed column ’GUBI’, because as mentioned only the active contracts which had no end date were kept. Last but not least, given the fact that there were multiple currencies (CNY, EUR, GBP, NOK, SEK, SKR, USD) converted all the prices into one currency EUR in this case, with index given the start date of the purchase contract. From this excel, the following columns were kept:

WKNR	TENR	LINR	GUVO	LIMG	BEPR
000	W700069S442K	10031	20091101	19500	31,7
009	996357	20719	20240101	0	15,22

Table 5: Six main columns of interest (copy of pc eu no name)

1) Starting of with the excel file called, ***copy of pc eu no name***. At this point, the creation of the general metrics started to be created. On the excel of ***copy of pc eu no name*** started by adding accordingly the Part’s Number ID and Category given from the excel ***facfasxppsparts***. Soon after that step, it was distinguished that for the same PN and supplier ID the price might have a great difference in one plant in comparison with other plants, for different reasons. One of them might be due to the different services. Check the following example:

WKNR	TENR	LINR	GUVO	LIMG	BEPR
000	992328	30372	20100101	0	123,58
000	992328	10094	20140301	18720	148,24
004	7400992328	20557	20171101	0	20,72
004	7400992328	10094	20240101	2217600	148,24

Table 6: Price Variability due to Supplier Services

In the table 6: row 3, it is shown that the price for PN 992328/7400992328 is 20,72 euro when normally it should be much higher, like the prices shown in the other lines. The reason of this special price is because supplier ID 20557 does not supply the parts but instead, it repacks it. Hence, 20,72 is the cost of the packaging and not for the part itself. That way, after contacting a colleague from the purchase team it was found out that given my previous excel, it could be distinguished 5 categories related to the PNs, with the following comments:

1. Coating
2. Interfacil

3. Packaging
4. Part in Dummy
5. Supplies Part

Having this information (from supplier name and purpose excel) for each supplier ID, it was given, the opportunity to delete all the data related to the first 4 categories and keep only the ones that 'supplied the part'. Furthermore, there were still some high prices in my data and for that reason out of the 15000+ lines of information, focused and explored the ones that the purchase price was more than 1000 euro, because there is needed a very long time to explore the ones of which the cost is less than 1000. That had as a result to check for more than 200 purchase contracts and verify if the prices I had in the excel were valid and verify.

WKNR	TENR	LINR	GUVO	LIMG	BEPR
005	21393278	20725	20220101	37380	701,85
005	21393278	10011	20230101	0	738,94
005	21393278	20343	20230101	196140	725,4
005	21393278	20386	20240101	0	74781,35

Table 7: Extreme invalid prices

Even after keeping only the suppliers that supply the parts, there seem to be still extreme high prices that are not valid and need to be deleted. For example the fourth line in the previous table.

During the data cleaning process, several issues with the dataset were identified, necessitating the removal of corresponding data lines. For instance, there were cases of extreme decreases in purchase prices over a short period. An example is PN 995006, where the price dropped from €26,781.90 (valid until 31/05/2023) to 6,277.51 euro (from 01/06/2024). Additionally, inconsistencies were observed between supplier IDs in the InforXpert system and those in the downloaded Excel file. For example, PN 7420925884 was associated with supplier ID 20710 in InforXpert but appeared with supplier ID 10011 in the Excel file.

Another issue involved discrepancies in pricing data between the two sources. For example, for PN W72007S450B at plant 001, the purchase price was listed as €612.073 in the Excel file but €707.26 in InforXpert for the same time period. Furthermore, some cases displayed purchase prices that exceeded sales prices, which should not occur; an example is PN 5003101266. Lastly, there were instances where part numbers (P/N) existed in the Excel dataset but were missing from InforXpert, such as PN 968753 at plant 005.

To address these issues, the corresponding data entries were deleted. This step was taken to ensure the dataset's quality and consistency for analysis. While this approach effectively eliminated problematic data, exploring alternative data cleaning methods, such as imputation strategies or deeper cross-referencing between datasets, could offer solutions for retaining more information in similar scenarios.

After all the deletions and the additions that were performed in the excel, got the following updated excel:

Plant	Part Number	Supplier ID	Supplier Name	Start Date	Price
004	7400997567	10011	Kamax GmbH & Co.KG	20230101	1191,9
004	7400997567	20343	Kamax GmbH & Co.KG (Osterode)	20230101	1331,14
004	7400997567	20725	KAMAX ALSF	20220101	1198,23
005	997567	10011	Kamax GmbH & Co.KG	20230101	1191,9
005	997567	20343	Kamax GmbH & Co.KG (Osterode)	20230101	1331,14
005	997567	20725	KAMAX ALSF	20220101	1198,23
008	997567	10011	Kamax GmbH & Co.KG	20230101	1191,9
008	997567	20343	Kamax GmbH & Co.KG (Osterode)	20230101	1331,14
008	997567	20725	KAMAX ALSF	20220101	1198,23
009	997567	10011	Kamax GmbH & Co.KG	20230101	1191,9
009	997567	20343	Kamax GmbH & Co.KG (Osterode)	20230101	1331,14
009	997567	20725	KAMAX ALSF	20220101	1198,23

Table 8: Table of Parts and Suppliers

Then, managed to transfer accordingly to each PN of that excel, their Facil PN ID and the Category of the fastening part.

3) Next step is to add the technical information. For that reason, created an individual excel for each Category, as the characteristics differ for nuts, bolts and screws. Hence, with the help of the three excels, ***Bolts***, ***Nuts*** and ***Screws***, given the ID no. that was added previously, transferred accordingly, the characteristics for each category. That had as result to have the following 3 metrics. One last comment is that, removed any column that seemed not needed as the values were the same throughout any row. These were, Firma and End Date (and Category for the individual excels).

Nut's Metrics includes: *Plant, Part Number, Facil PN ID, Supplier ID, Supplier Name, Start Date, Oil (Price),(Delivery) Quantity, (Purchase) Price, Thread Type, Thread Size, Thread Pitch, Type, Diameter, Height, Property Class, Tool Size, Locking Feature, Locking Type, Has Paint Clearing, Has Washer, Washer, Washer Diameter, Washer Thickness, Weight.*

Screw's Metrics includes: *Plant, Part Number, Facil PN ID, Supplier ID, Supplier Name, Start Date, Oil (Price), (Delivery) Quantity, (Purchase) Price, Thread Type, Thread Size, Head, Head Diameter, Head, Height, Thread Special, Tool Size, Tool, Total Length, Locking Feature, Locking Type, Thread Point, Has Washer, Washer, Washer Diameter, Washer Thickness, Weight.*

Bolt's Metrics includes: *Plant, Part Number, Facil PN ID, Supplier ID, Supplier Name, Start Date, Oil (Price),(Delivery) Quantity, (Purchase) Price, Thread Type, Thread Size, Thread Pitch, Head, Head Diameter, Head Height, Length, Property Class, Thread Length, Thread Special, Tool Size, Tool, Total Length, Has Lock, Locking Type, Has Paint Clearing, Has Pilot, Pilot, Has Shoulder, Shoulder Diameter, Shoulder Height, TF, Has Washer, Washer, Washer Diameter, Washer Height, Has Washer, Weight.*

4) Last but not least the oil price of the market was added, converted to euro based on start date of the purchase contract. The oil prices have been added accordingly in the previous three excels (Bolts, Nuts, Screws). Finally, the main focus will be on active latest purchase prices with earliest starting date, that of 01/01/2021. As in the previous period of time, after communication with the sales manager, the data lacks reliability.

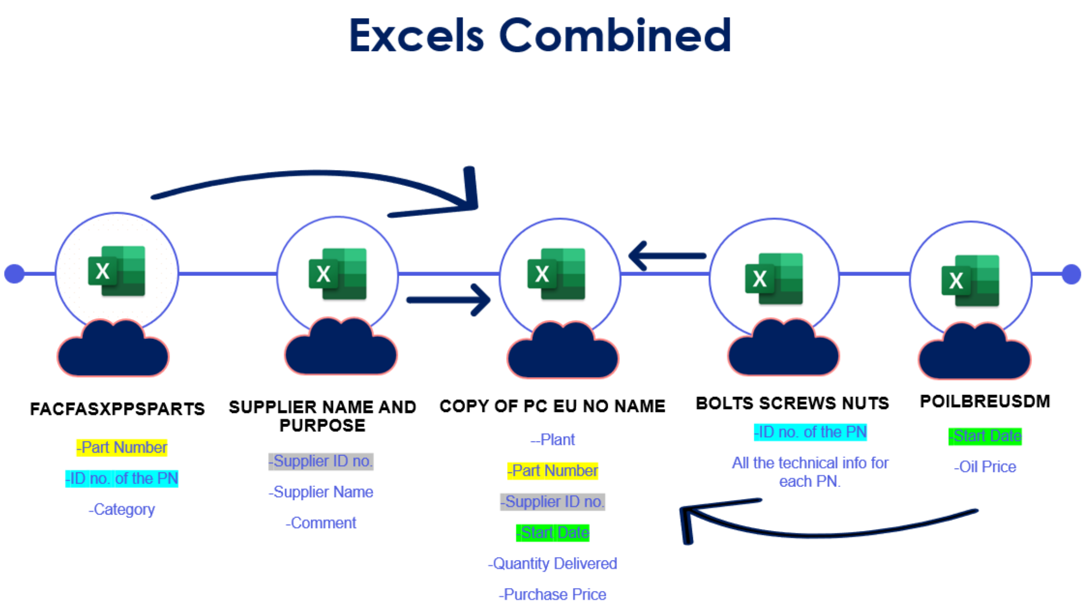

Figure 2: Excels Combination

4.3 Exploratory Data Analysis

Exploratory data analysis (EDA) is an essential stage in the data analysis process. Its objective is to examine data sets and describe their essential properties, which is commonly done visually. It helps to comprehend the data, discover patterns, spot abnormalities, and check assumptions. It uses a variety of statistical graphics and visualization tools, including histograms, box plots, scatter plots, and bar charts. Common descriptive statistics include mean, median, mode, variance, and standard deviation. [**3**]. Below, you will discover part of the visualization done through 'lookerstudio' [**4**]. The rest of the graphs will be found on the appendix and are created via jupyter notebook python version 3.10.For more details, see Appendix 12.

4.3.1 Nuts Data Overview

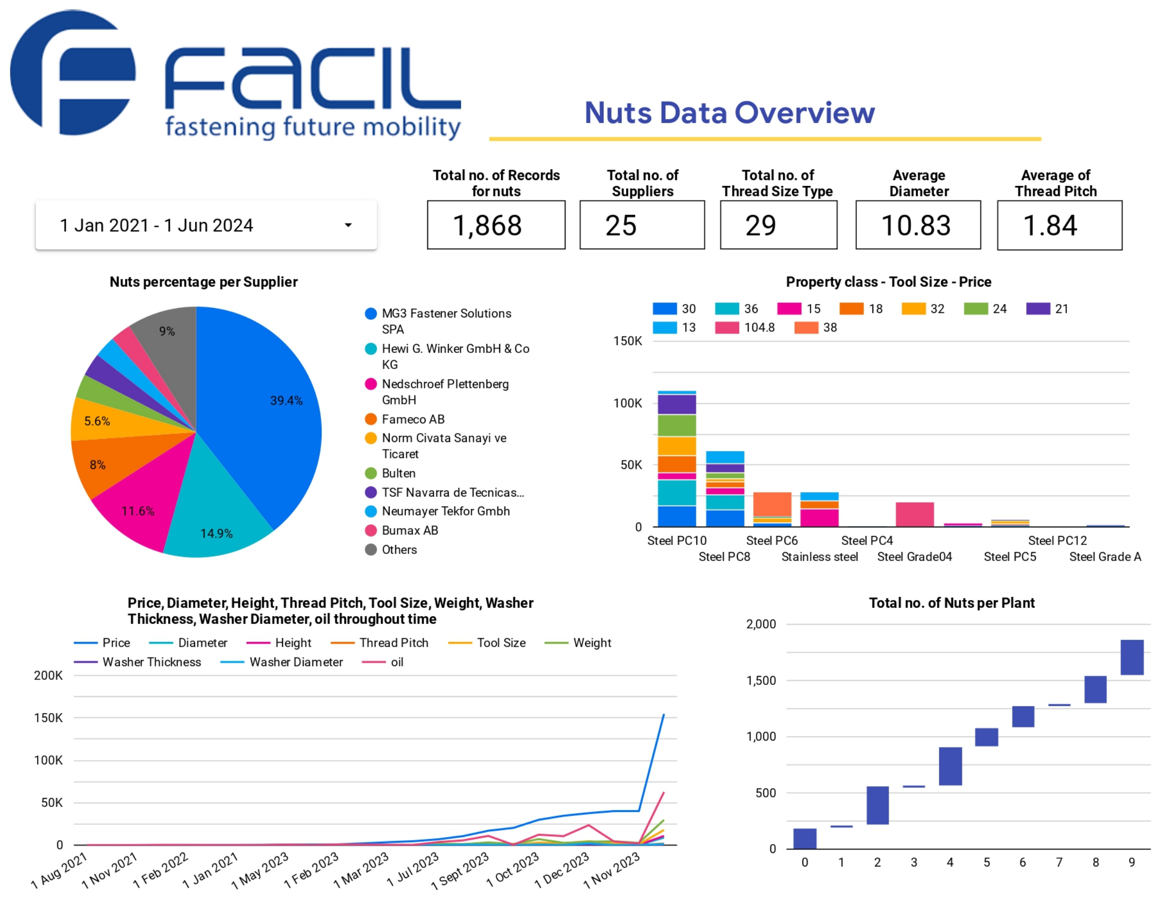

Figure 3: Nuts Data Visualization

In this slide, we see a review regarding the Nuts data for the period of time 1st of January 2021 till 1st of June 2024. The total no. of the PNs that are nuts is 1868 parts. For these 1868 parts there are in total 25 suppliers that supply them. In the pie-chart it is shown the percentage of nuts per supplier with most popular one being MG3 Fastener Solutions SPA. In the chart of Property class, tool size and price, we can see the price per property class by having as breakdown dimension the tool size per property class. In the second chart multiple independent variables have been added to investigate the relationship between them and the dependent variable, the purchase price throughout time. Lastly, the final chart shows the total number of nut records counted individually for each plant.

4.3.2 Screws Data Overview

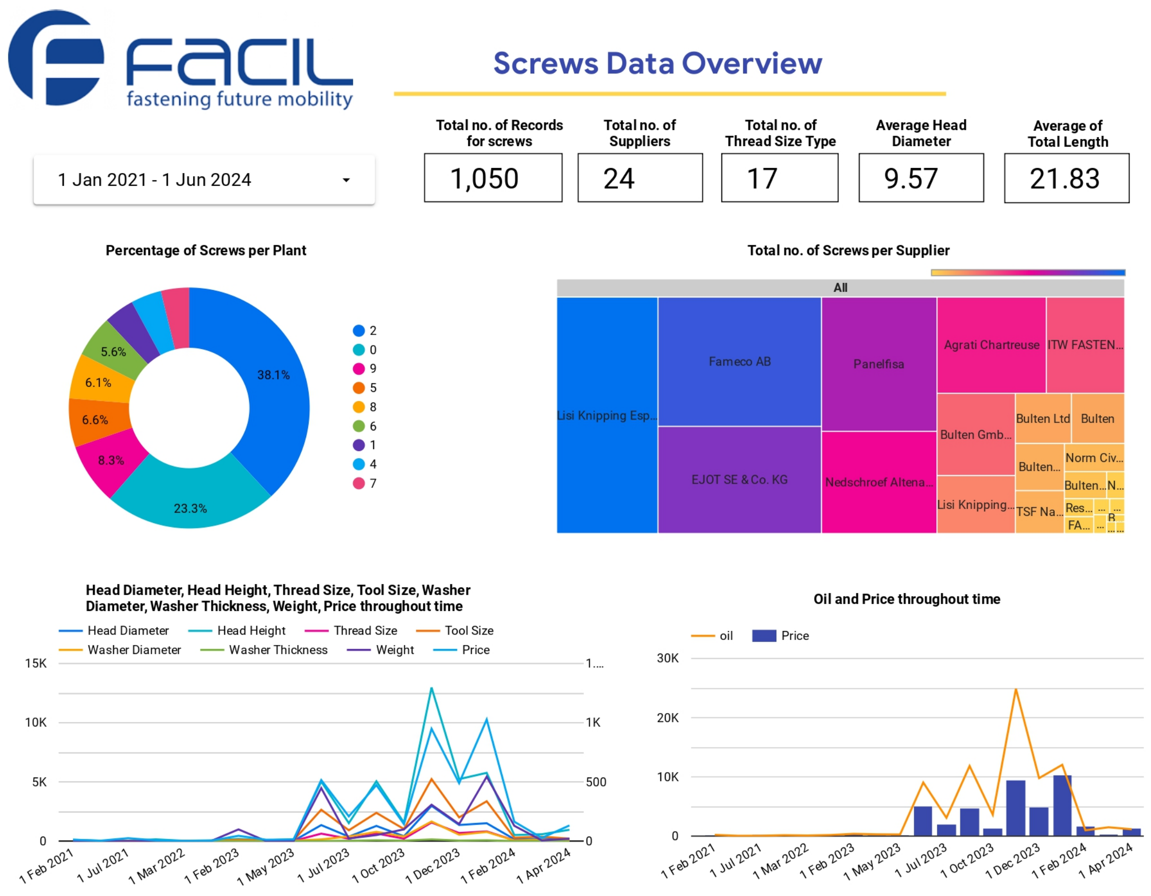

Figure 4: Screws Data Visualization

In this slide, we see a review regarding the Screws data for the period of time 1st of January 2021 till 1st of June 2024. The total no. of the PNs that are nuts is 1050 parts. For these 1050 parts there are in total 24 suppliers that supply them. In the pie-chart it is shown the total number of screws records observed per plant. The second chart visualizes the total number of screws records per supplier and the most recognizable one is Lisi Knipping Espana Sa. For the following graph, same logic as in the nuts data, so in the screws date multiple variables such Head Diameter, Head Height, Thread Size, Tool Size, Washer Diameter, Washer Thickness, Weight were added with the purpose to see their relationship with the purchase price throughout time. As for the last chart, oil price is the only external variable and was interesting to check its relationship individually with the purchase price.

4.3.3 Bolts Data Overview

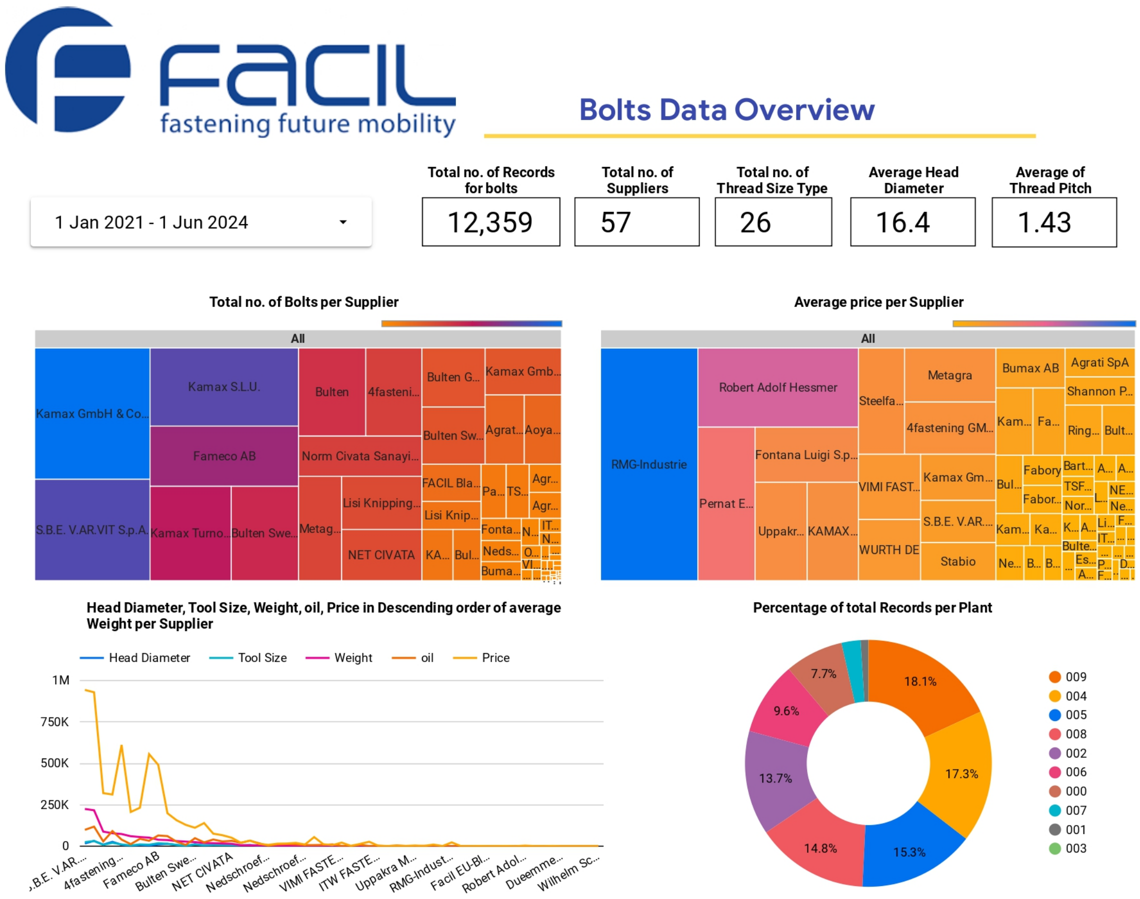

Figure 5: Bolts Data Visualization

In this slide, we see a review regarding the Bolts data for the period of time 1st of January 2021 till 1st of June 2024. The total no. of the PNs that are nuts is 12359 parts. For these 12359 parts there are in total 57 suppliers that supply them. The rest is information given the bolts data. In specific, in the first chart it is obvious that the most popular supplier given the total number of bolts, it is Kamax GmbH and Co GK. The second chart introduces a more innovative approach by calculating the average price per supplier and this time even though supplier Kamax GmbH and Co GK provides the most bolts, the supplier with the highest average price is supplier RMG-Industrie. In the third graph, we see an harmony regarding the relationship of the purchase price and the Head Diameter, Tool Size, Weight, oil, in Descending order of average Weight. Last but not least, in the pie chart it is shown the percentage of total records regarding bolts for each plant.

4.4 Summary Statistics

In this section we will get a more statistical background of the three datasets for Nuts, Screws and Bolts (as the features differ for these three categories of fastening type).Additionally, the features that had the most impact on the price were identified, and outliers were removed accordingly (after communication).

4.4.1 Summary Statistics for the price of Screws

In the following table, is emphasized the statistical summary table from Screws dataset, for the dependent variable of our interest the 'Price'.

Table 9: Summary Statistics for Price

Statistic	Value
Count	1050
Mean	40.49
Standard Deviation	57.81
Minimum	5.52
25th Percentile	18.14
Median (50th Percentile)	28.57
75th Percentile	40.9
Maximum	794.91

After the removal of the outliers the new statistical results were adjusted as followed:

Table 10: Screws Summary Statistics for Price: Without Outliers

Statistic	Value
Count	972
Mean	29.25
Standard Deviation	14.84
Minimum	5.52
25th Percentile	17.6
Median (50th Percentile)	26.81
75th Percentile	37.85
Maximum	74.73

The outliers removal had as result, to get a smaller mean and standard deviation in the screws dataset. As for the decrease of the data, 78 rows of outliers, were not taken into account.

4.4.2 Summary Statistics for the price of Nuts

In the following table, is emphasized the statistical summary table from Nuts dataset, for the dependent variable of our interest the 'Price'.

Table 11: Summary Statistics for Price

Statistic	Value
Count	1868
Mean	217.01
Standard Deviation	573.32
Minimum	1.08
25th Percentile	33.19
Median (50th Percentile)	75.8
75th Percentile	176.52
Maximum	10038.38

What is worth mentioning, is the extreme Price difference between the minimum noted price with the maximum price noted. As well, the deviation is much larger in comparison with the dataset of screws. After the removal of the outliers the new statistical results were adjusted as shown:

Table 12: Nuts Summary Statistics for Price: Without Outliers

Statistic	Value
Count	1647
Mean	90.5
Standard Deviation	84.17
Minimum	1.08
25th Percentile	30.62
Median (50th Percentile)	63.7
75th Percentile	119.08
Maximum	388.86

The outliers removal had as result, to get a much smaller mean and standard deviation in the screws dataset. As for the decrease of the data, 221 rows of outliers, were not taken into account.

4.4.3 Summary Statistics for the price of Bolts

In the following table, is emphasized the statistical summary table from Bolts dataset, for the dependent variable of our interest the 'Price'.

Table 13: Summary Statistics for Price

Statistic	Value
Count	12359
Mean	473.39
Standard Deviation	1755.62
Minimum	4.8
25th Percentile	68.2
Median (50th Percentile)	170.56
75th Percentile	445.62
Maximum	153908.81

What is worth mentioning, expect the extreme Price difference between the minimum noted price with the maximum price noted. As well, the deviation is much larger and the records are 10times more, in comparison with the other two datasets. After the removal of the outliers the new statistical results were adjusted as followed:

Table 14: Bolts Summary Statistics for Price: Without outliers

Statistic	Value
Count	11009
Mean	228.90
Standard Deviation	227.83
Minimum	4.8
25th Percentile	61.85
Median (50th Percentile)	139.75
75th Percentile	312.9
Maximum	1008.4

The outliers removal had as result, to get a much smaller mean and standard deviation in the screws dataset. As for the decrease of the data, 1350 rows of outliers, were not taken into account.

4.5 Missing Data

There are numerous approaches to dealing with missing data, and the choice of method depends on the nature of your research and the importance of the missing information. Common methods to handle missing data include:

1) Imputation

2) Deletion

3) Model-Based Method

4) Analysis-Specific Handling.[**6**]

In this case, mean imputation was applied by replacing the missing values with the mean of the respective column. This approach assumes that the data are missing at random and that the mean adequately represents the central tendency of the column. The dataset consists of 'fastening part's price'. While mean imputation can reduce the bias introduced by missing data, it may oversimplify variability and relationships in the data. Further analysis would assess whether this choice is appropriate for the specific context of this research.

In specific for the given three data sets, please refer on the following table:

Missing Data in Nuts	
Locking Type	992
Washer	1589
Missing Data in Screws	
Locking Type	1007
Washer	741
Missing Data in Bolts	
Tool	25
Locking Type	11042
Pilot	9402
Washer	10882

Table 15: Summary of Missing Data

4.6 Feature Selection

A critical stage in the machine learning process is feature selection, which is picking the variables or features from your dataset that are most pertinent. Feature selection aims to reduce over fitting, enhance model performance, and facilitate model interpretation. Here are a few crucial elements [7]:

- Dimensionality Reduction
- Improved Model Performance
- Overfitting Prevention
- Interpretability

For all the three data sets of Nuts, Screws and Bolts the top 10 most influential characteristics were noted for each category. In the following graph it is shown for example that Total Length seems to have the biggest influence in comparison with other technical variables regarding bolts, nuts and screws. Weight is in the top 3 factors for the three datasets (bolts, nuts and screws) as for length and tool size are included as well as the top common factors.

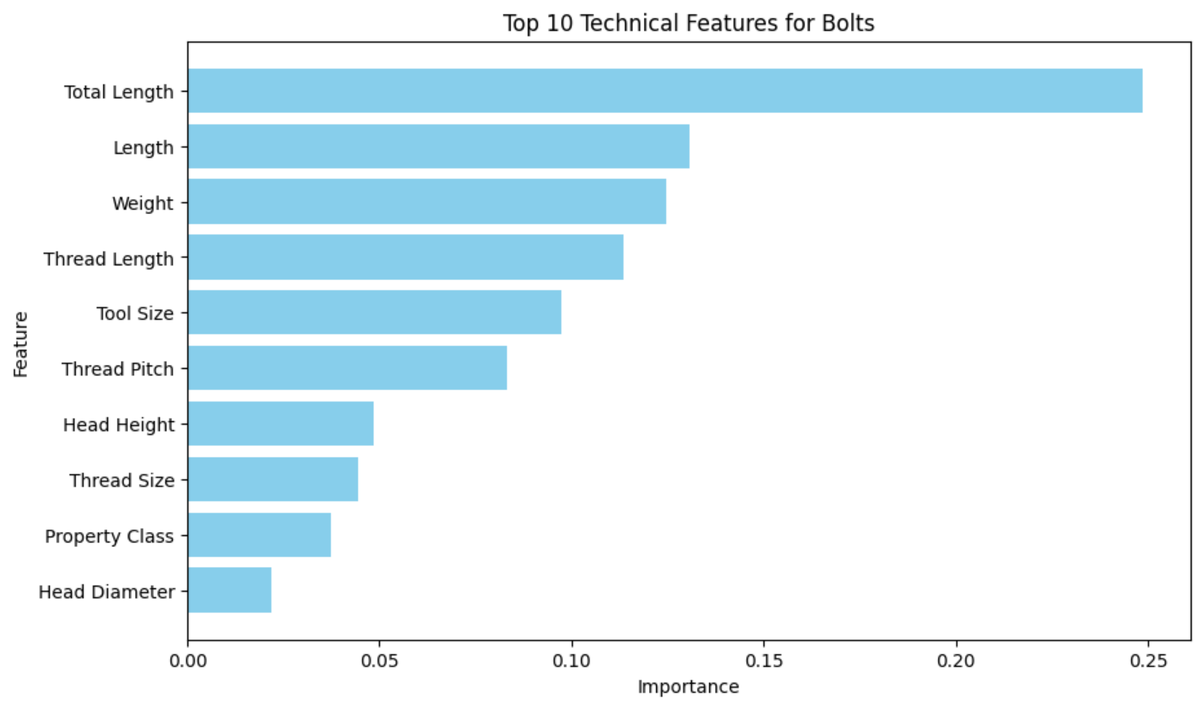

Figure 6: Top 10 Technical Features for Bolts

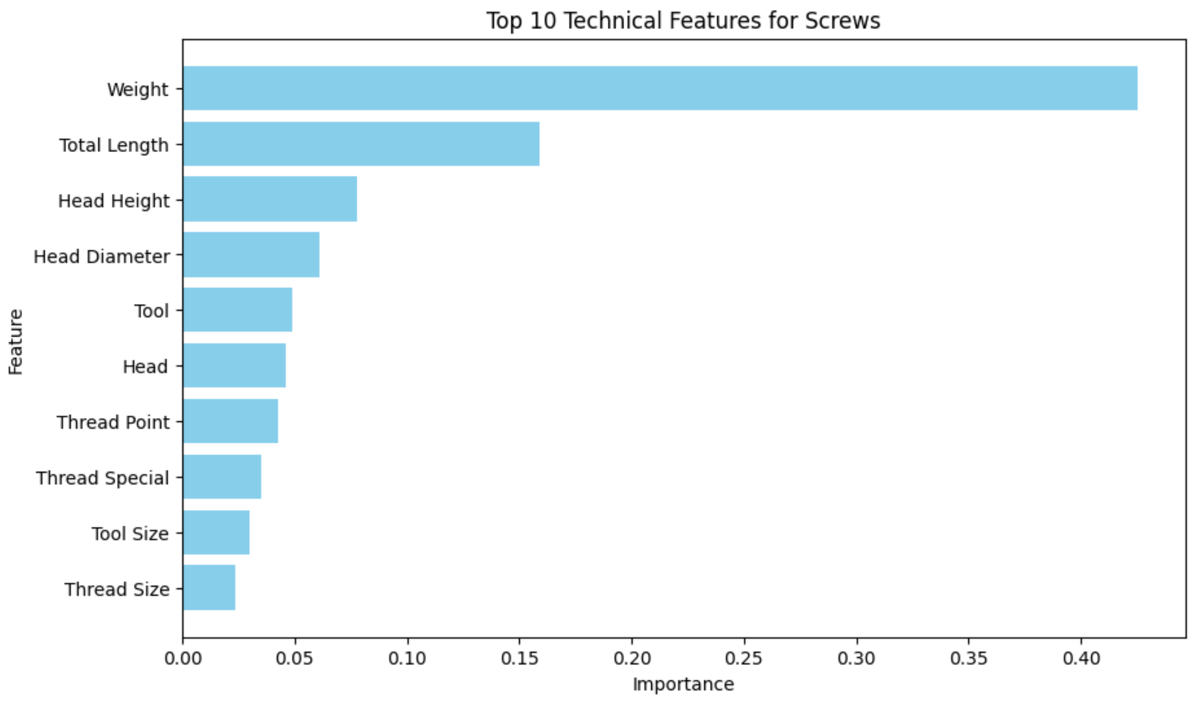

Figure 7: Top 10 Technical Features for Screws

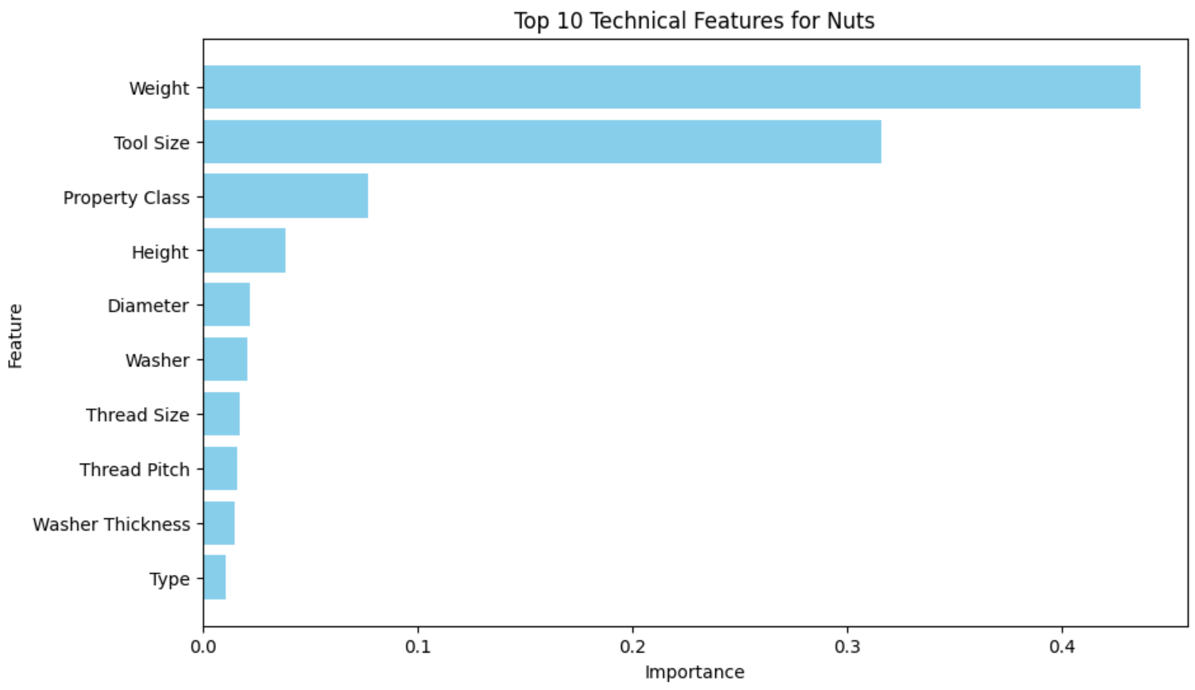

Figure 8: Top 10 Technical Features for Nuts

Additionally, it could not be missed a color-coded heatmap that shows the correlation matrix between the variables, making it easier to see and understand the connections and dependencies inside the dataset.

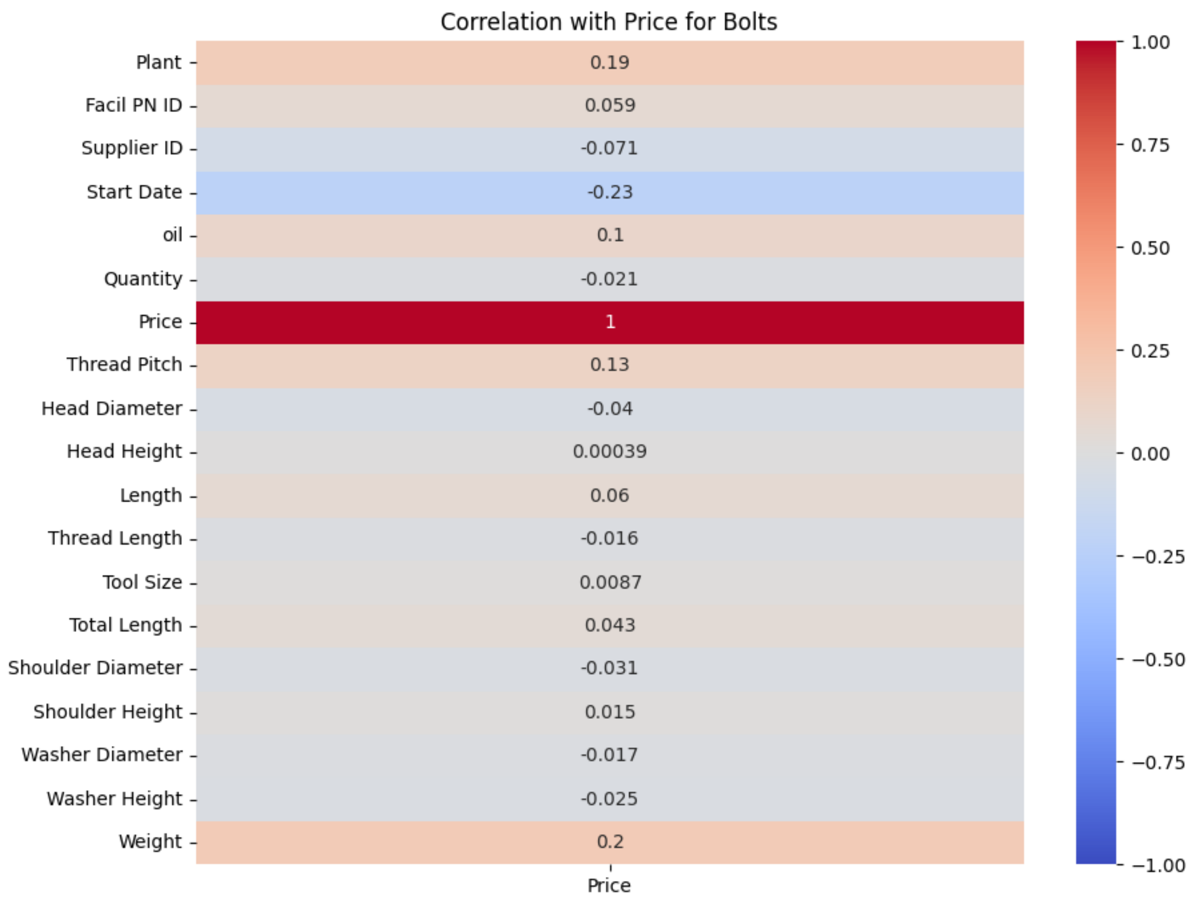

Figure 9: Correlation Matrix of Bolts

Bolts

Total Length: Positive correlation (0.043). Indicates that as total length increases, the price of bolts increases slightly.

Weight: Positive correlation (0.2). Suggests that heavier bolts are more expensive.

Thread Pitch: Moderate positive correlation (0.13). Suggests that bolts with a finer or coarser thread pitch might have an effect on pricing.

Tool Size: Very low positive correlation (0.0087). Tool size is not a significant factor in pricing.

Variables such as washer dimensions and shoulder height have negligible correlations.

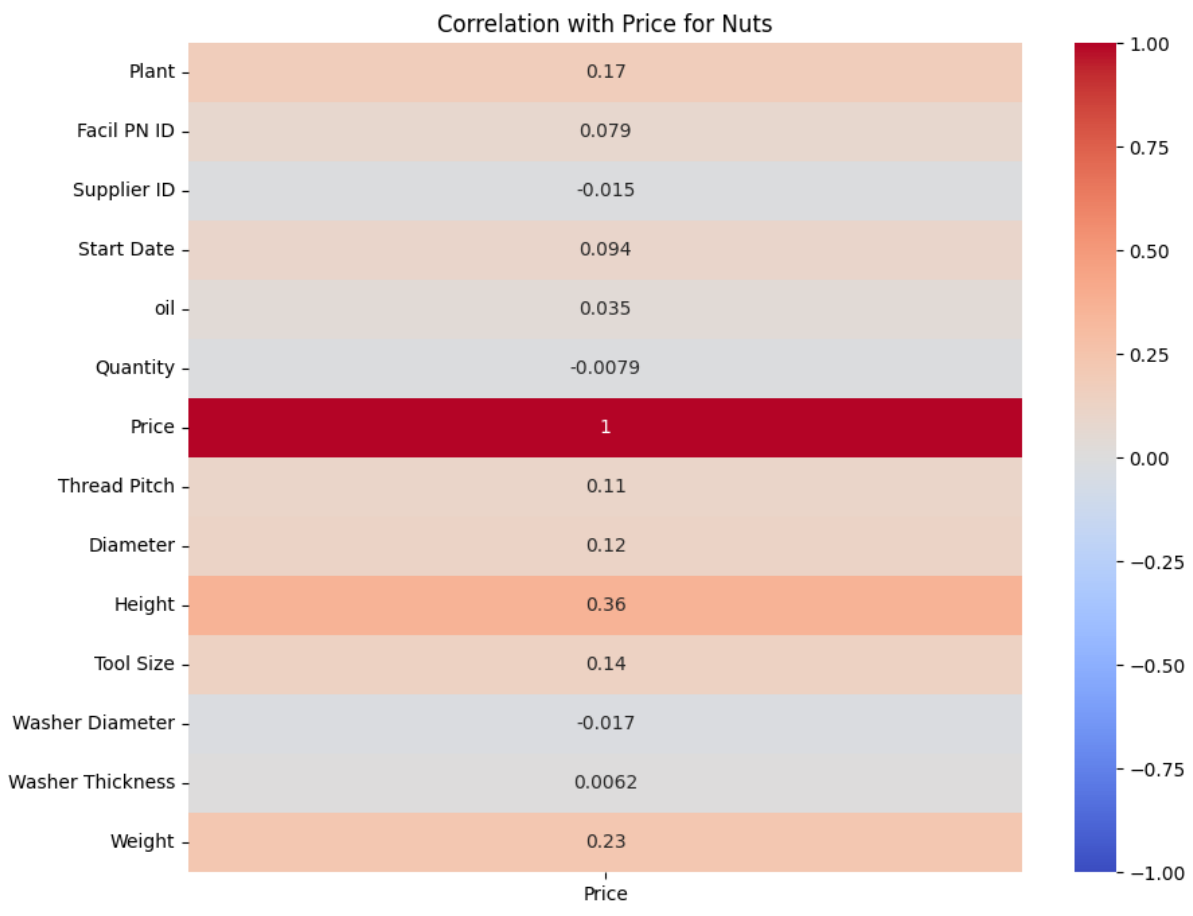

Figure 10: Correlation Matrix of Nuts

Nuts

Height: Strongest positive correlation (0.36). Indicates that taller nuts tend to be more expensive.

Weight: Positive correlation (0.23). Heavier nuts are also more expensive, but to a lesser degree than bolts.

Tool Size: Moderate positive correlation (0.14). Suggests an effect of the required tool size on nut pricing.

Thread Pitch and Diameter: Weak positive correlation (0.11–0.12). These features play a minor role in price.

Features such as washer dimensions and quantities show negligible correlation with pricing.

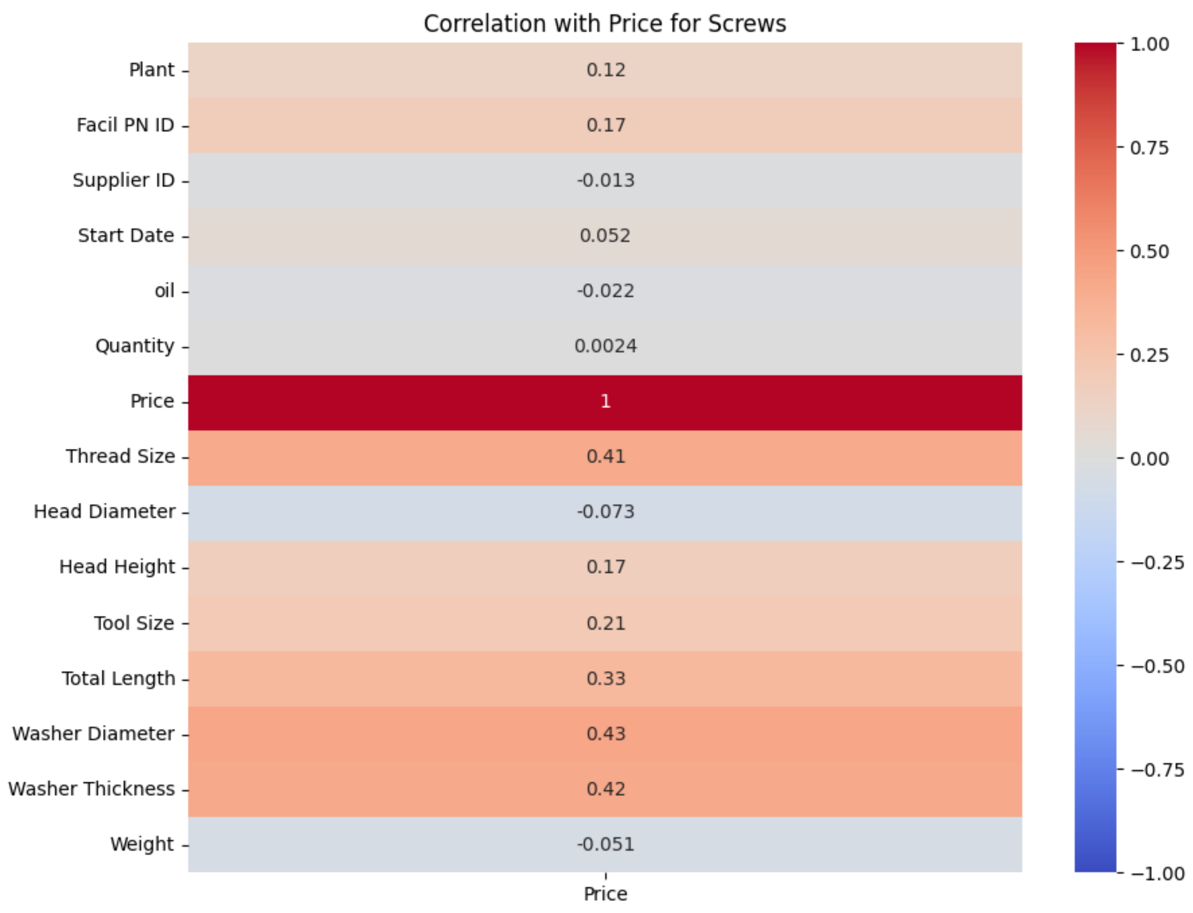

Figure 11: Correlation Matrix of Screws

Screws

Thread Size: Strong positive correlation (0.41). Suggests that screws with larger threads are more expensive.

Washer Diameter and Washer Thickness: Moderate to strong correlations (0.42–0.43). Indicates significant impact on screw pricing.

Total Length: Positive correlation (0.33). Longer screws tend to be more expensive.

Head Height: Moderate positive correlation (0.17). The height of the head also has a noticeable effect on price.

Weight: Positively correlated for bolts and nuts but negatively correlated for screws, showing different design or pricing logic between product categories.

Dimensions: Features like height, length, and thread size/dimensions strongly affect the price across all three categories. Larger products in these dimensions are generally more expensive.

Washer-Related Dimensions (Screws): Washer diameter and thickness show strong pricing influence on screws compared to bolts and nuts, highlighting product-specific importance.

5 Methodology

Methodologically it was assessed a broad spectrum of machine learning models for the predictive modeling challenge in the first stage of the model development. Among the potential algorithms were Gradient Boosting, Random Forest, and KNN.

5.1 Model 1: KNN

Data Preparation:

For the screws, nuts, and bolts datasets, the following preprocessing steps were applied:
Feature Selection: Only relevant features were retained for each category (e.g., Length, Weight, Tool Size).
Outlier Removal: Extreme values were removed to improve model robustness.
One-Hot Encoding: Categorical features were converted into numerical format.
Missing Value Imputation: Missing values were filled using mean imputation.
Data Splitting: Each dataset was divided into training (80%) and testing (20%) sets.

K-Nearest Neighbors (KNN) Training

Model Selection:

A K-Nearest Neighbors (KNN) Regressor was chosen for its simplicity and effectiveness in capturing local relationships within data.

Hyperparameter Optimization:

The number of neighbors (k) was tuned by iterating over values from 1 to 20. For each k, the model was trained and tested, and the Root Mean Squared Error (RMSE) was computed.
Best K for bolts: 2
Best K for nuts: 2
Best K for screws: 3

5.2 Model 2: Random Forest

Data Preparation:

For the screws, nuts, and bolts datasets, the following preprocessing steps were performed:
Outlier Removal: Cleaned the data by removing extreme values that could impact model performance.
Feature Encoding: Transformed categorical features into numerical format using one-hot en-

coding.

Handling Missing Values: Imputed missing values using the mean to ensure a complete dataset.

Data Splitting: Divided the data into training (80%) and testing (20%) subsets for independent evaluation.

Random Forest Training

Model Selection:

A Random Forest Regressor was chosen for its ensemble-based approach, which combines multiple decision trees to improve prediction accuracy and robustness.

Hyperparameter Tuning:

A grid search with 5-fold cross-validation was used to optimize the model's hyperparameters. The parameter grid included: n_estimators: Number of trees in the forest (100, 200, 500). max_depth: Maximum depth of the trees (None, 10, 20). min_samples_split: Minimum number of samples required to split a node (2, 5, 10). min_samples_leaf: Minimum number of samples required at a leaf node (1, 2, 4).

Cross-Validation:

During the grid search, 5-fold cross-validation was performed to evaluate the performance of different hyperparameter combinations. The best model was selected based on the lowest mean squared error (MSE) across validation folds.

5.3 Model 3: Gradient Boosting

Data Preparation:

The dataset for screws, nuts, and bolts was prepared using the following steps:

Outlier Removal: Ensured the data was clean and free from extreme values that could skew model performance.

Feature Encoding: One-hot encoding was applied to categorical features, converting them into numerical form suitable for machine learning algorithms.

Handling Missing Values: Missing data was imputed using the mean value of each feature to maintain data integrity.

Data Splitting: The dataset was split into training (80%) and testing (20%) subsets for independent evaluation of the model.

Gradient Boosting Training

Model Selection:

A Gradient Boosting Regressor was chosen for its ability to handle regression tasks through iterative tree-based learning.

Hyperparameter Tuning:

A grid search was conducted to identify the best combination of hyperparameters.

The parameter grid included: n_estimators: Number of trees (100, 200, 500). learning_rate: Shrinkage parameter to control step size (0.01, 0.1, 0.2). max_depth: Depth of individual trees (3, 4, 5). min_samples_split: Minimum samples required to split a node (2, 5, 10). min_samples_leaf: Minimum samples in a leaf node (1, 2, 4).

Cross-Validation:

A 5-fold cross-validation was applied during the grid search.

The best model was selected based on the lowest mean squared error across folds.

Cross-validation results were summarized as mean and standard deviation of RMSE.

6 Model Performance Evaluation

Evaluating machine learning models is crucial to ensure their reliability, enable precise comparisons, and identify areas for improvement. Key components in model evaluation include:

6.1 Cross Validation

Cross-validation provides a robust estimate of model performance by dividing the data into multiple folds. The model is trained on a combination of these folds and tested on the remaining fold. This process is repeated across all folds, ensuring every data point is used for both training and testing. CV reduces the risk of overfitting to a particular train-test split and offers a more generalized evaluation metric.

6.2 Hyper parameter Tuning

Hyperparameters are model parameters set prior to training and have a significant impact on model performance. For instance, in the KNN Regressor, the number of neighbors (k) is the primary hyperparameter. We tuned k by iterating through values from 1 to 20 and selected the value that minimized the Root Mean Squared Error (RMSE). HPO helps balance model complexity and prediction accuracy. [8]

6.3 Root Mean Square

The RMSE is used to measure the average magnitude of discrepancies between actual and predicted values:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Lower RMSE values indicate better model performance.

7 Results

After training and evaluating the models, the following table presents the Root Mean Squared Error (RMSE) values for each approach, summarizing the comparison of the fitted models.

	RMSE for bolts	**RMSE for nuts**	**RMSE for screws**
KNN	93,25	33,76	8,21
Random Forest	77,95	28,64	6,77
Gradient Boosting	77,17	29,06	6,86

Table 16: RMSE values for different models and components

As we can confirm the differences between the results ain't that big. Still the model with the best results seems to be given from Random Forest model. The highest deviation is noted on the category of bolts and least deviation is noted on the category of screws. In this case we can come to the conclusion, that despite the fact that the category of bolts had the highest number of observations it seems that, the cause could be the higher number varieties in comparison with the category of screws or the data set has had more pollution, or maybe both of these two options.

Important to mention at this point, is the Python's libraries used in the code for training and evaluation are:

Scikit-learn (sklearn):

RandomForestRegressor and GradientBoostingRegressor: Used for the model training and predictions with Random Forest and Gradient Boosting algorithms. GridSearchCV: Used for hyperparameter tuning via grid search to find the best model parameters. cross_val_score: Used for cross-validation of models. mean_squared_error: Used to calculate the root mean squared error (RMSE) for evaluating model performance.

Pandas (pd):

Used for handling data, creating DataFrames for storing actual and predicted values, and manipulating data for evaluation.

Matplotlib (plt):

Used for creating scatter plots and figures for visualizing the comparison between actual and predicted values.

Seaborn (sns):

Used for creating violin plots to visualize the distribution of actual vs predicted values.
These libraries play a vital role in model building, training, hyperparameter tuning, validation, evaluation, and visualization.

8 Prediction

The relationship between the predicted purchase price and the actual price, illustrated for all three models (Gradient Boosting, Random Forest, and KNN), provides important insights into the strengths and limitations of each approach:

Strengths: Random Forest Dominance: Among the three models, Random Forest consistently demonstrates superior predictive accuracy, as evidenced by the tight clustering of data points around the diagonal line across all categories (Bolts, Nuts, Screws). This indicates its strong ability to capture the underlying relationships between features and target prices.

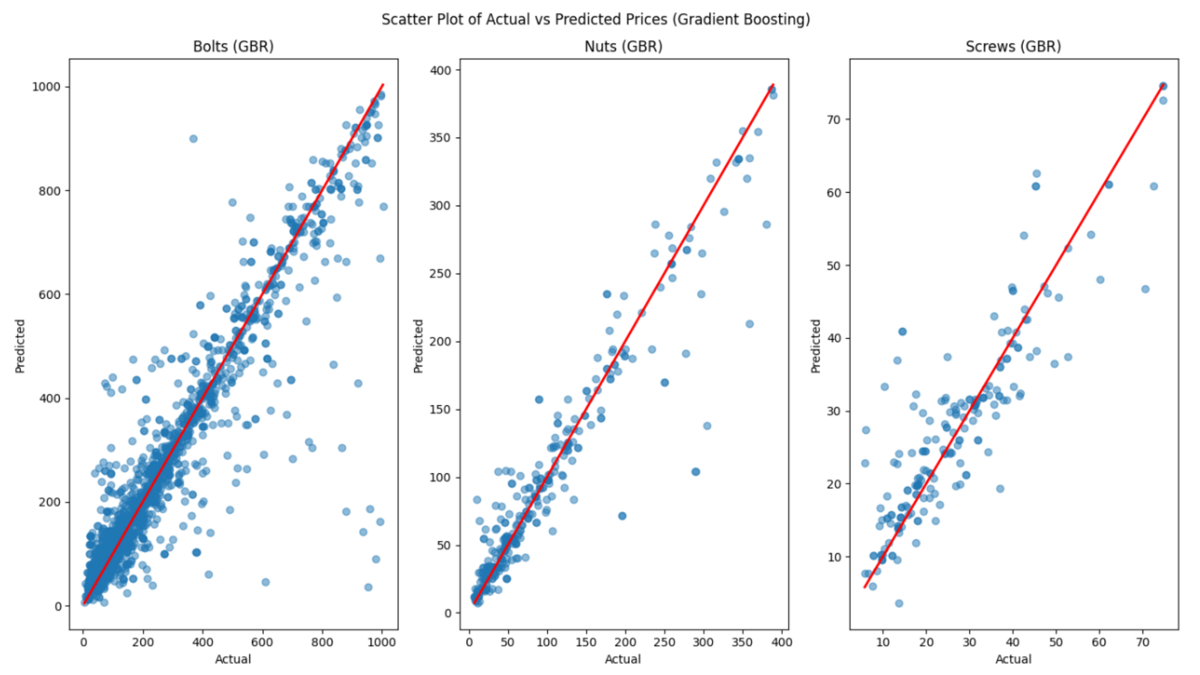

Figure 12: Scatter Plot of Actual vs Predicted Prices (Gradient Boosting)

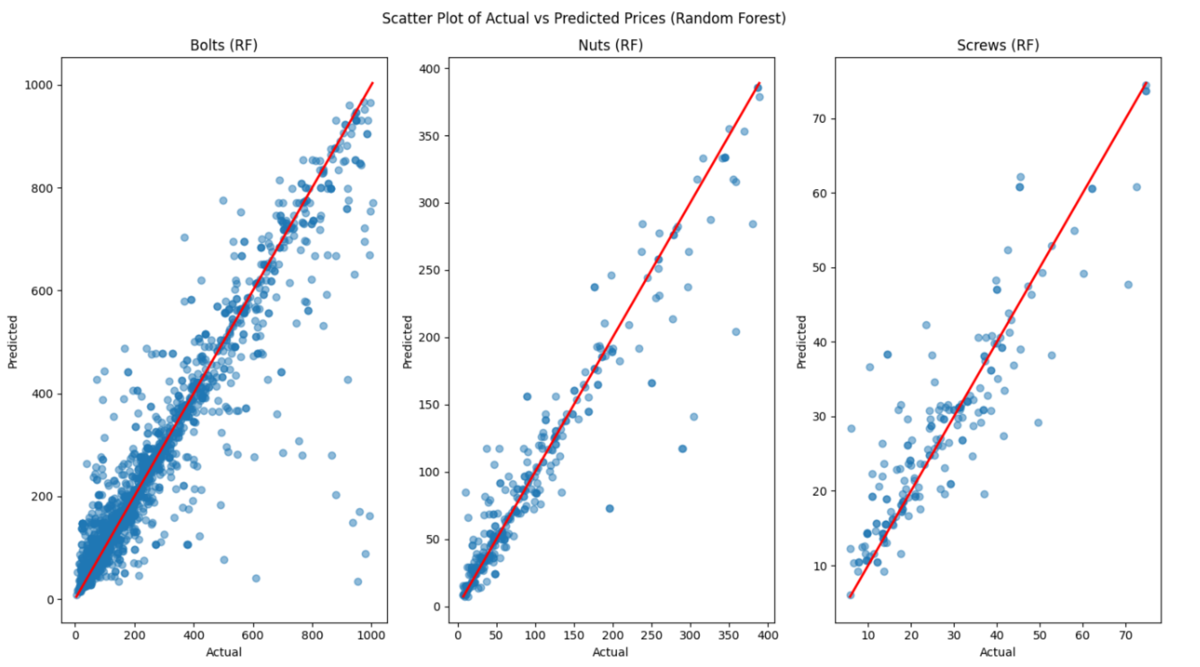

Figure 13: Scatter Plot of Actual vs Predicted Prices (Random Forest)

General Consistency Across Categories: The uniformity in performance across bolts, nuts, and screws suggests that the selected features—such as weight, thread size, and head height—are well-suited to explain price variation across these product types.

Limitations and Observations: Increased Variability at Higher Price Points: All models show greater dispersion of points for higher actual price values, indicating reduced predictive accur-

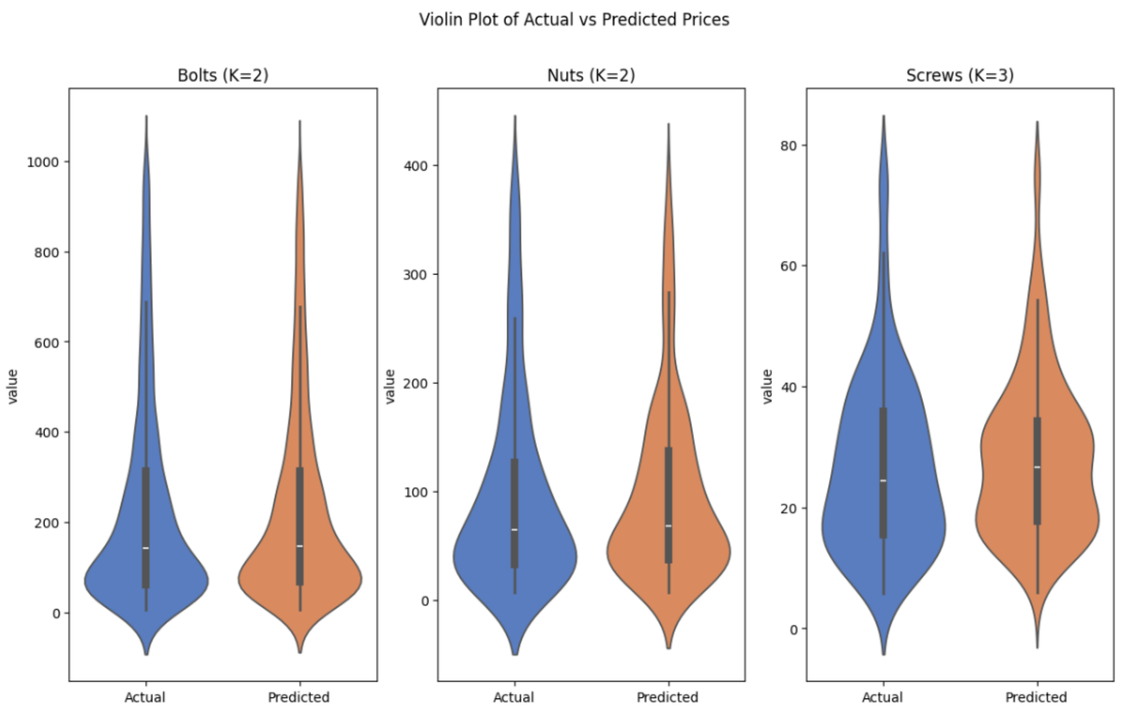

Figure 14: Scatter Plot of Actual vs Predicted Prices (KNN)

acy for these cases. This trend could stem from overfitting to mid-range price samples or a lack of sufficient training data for extreme price values.

Outliers and Residual Patterns: The scatter plots reveal several outliers, particularly in the Gradient Boosting and Random Forest models. These anomalies may indicate unmodeled data nuances, misrepresentative samples and noise in the dataset.

KNN Model Limitations: The violin plots for KNN highlight significant discrepancies between actual and predicted prices, particularly for bolts and nuts. This suggests that KNN struggles to generalize effectively, likely due to its sensitivity to local patterns and the choice of hyperparameters.

Potential Feature Redundancy: While feature selection appears effective overall, the comparable performance across bolts, nuts, and screws could suggest some redundancy or limited feature-specific insights, especially in cases where variability is high.

9 Challenges

In a this master thesis multiple challenges occurred through out it. Starting by external point of view, receiving all the needed data and making clear on which to focus on, was really challenging. Probably this was caused due to poor understanding as communication is the key for everything, specially when it is repetitive. That could be solved by weekly meetings with Facil and by being in that way active, more things could become clear on time and better analysis could occur.

Misunderstandings as well seem to have occurred on the hours of focusing in this project in the company, which this led to limited time to really be able to focus. In more detail, time has been really crucial, as a lot of situations took place and what could be improved regarding this analysis, would be regarding the technique of its analysis, which means that instead of using data points we could use longitudinal perspective, as it is a procedure of continually watching or measuring the same subjects over an extended period of time, which can range from many years to decades. This type of research is very important for comprehending long-term consequences, changes, and advancements.

From practical perspective, the data is mainly not clear which makes the predictions more poor given the InforXpert system. Moving to SAP system, more valid data will be selected, which this is followed by better predictions and analysis.

After data clearance and excel combination, unfortunately the elaborated work had been vanished and had to start all over again, which takes time to elaborate again the formulas.

10 Ethics

Ethics form a cornerstone of research and professional integrity, ensuring that the outcomes of this work adhere to the highest moral and societal standards. This thesis, which focuses on the cost prediction of automotive fastening parts, was guided by a commitment to ethical practices across every stage of its development.

1. Data Privacy and Confidentiality

Data used in this study were obtained from Facil Company and related departments. Every effort was made to ensure that sensitive information, such as supplier details or proprietary technical specifications, was handled responsibly. To maintain confidentiality, datasets were anonymized where required, and access to sensitive information was restricted to authorized personnel.

2. Accuracy and Honesty in Research

This study strives to present all findings with full transparency and objectivity. Manipulation or misrepresentation of data and results was strictly avoided to maintain the credibility of the research. Outlier management and data imputation methods were disclosed and implemented with caution to ensure their compatibility with the integrity of the analysis.

3. Respect for Stakeholders

Throughout the research, the interests and reputations of stakeholders, including Facil and associated personnel, were carefully considered. Decisions regarding data cleaning, feature selection, and modeling were made to maximize the value provided to stakeholders while minimizing potential risks arising from inaccurate or misleading insights.

4. Compliance with Legal and Organizational Standards

The research adhered to all legal regulations concerning data usage and adhered to the internal guidelines set by Facil. This included using licensed software, citing sources appropriately, and ensuring ethical collaborations with involved departments.

5. Social Responsibility

The broader implications of the research were considered in terms of its potential impact on cost efficiency, resource allocation, and fairness in industrial decision-making. Ethical forecasting methods were attempted to be employed to avoid speculative conclusions.

11 Conclusion

The analysis highlights several key findings and areas for improvement in predicting purchase prices for bolts, nuts, and screws:

Model Performance: Among the evaluated models, Random Forest demonstrated the strongest predictive performance across all categories, while KNN showed limitations, particularly in handling global patterns. Gradient Boosting offered competitive results but showed some susceptibility to outliers.

Variability and Outliers: Predictions for higher price ranges exhibited greater variability, suggesting limitations in capturing extreme values effectively. Additionally, outliers in the predictions point to unmodeled patterns or noisy data, which require further investigation.

Data Quality Challenges: Despite improvements in the dataset, much of the data remains outdated, as it is based on older part numbers. This has a significant impact on model accuracy and limits the insights that can be drawn for new or uncommon products.

Based on the findings, the following actions are recommended:

Error Analysis: Investigate the outliers by reviewing feature values associated with those predictions. This can help identify unstructured patterns or inconsistencies in the data.

Enhancing Model Performance: Explore feature engineering techniques to create new variables or interactions that better capture price variability. Test alternative models or ensemble methods (e.g., combining Random Forest with Gradient Boosting) to improve predictive accuracy. Adjust hyperparameters (e.g., increasing the number of trees in Random Forest) or apply specialized techniques such as quantile regression for price ranges with high variability.

Data Cleaning and Maintenance: Establish a dedicated team to systematically clean and update the dataset, prioritizing more recent data while progressively addressing older records. Continuously add clean, structured information to the SAP system to ensure data relevance and accuracy. Without such efforts, the current issues of data inconsistency and high deviations are likely to persist.

By implementing these recommendations, the organization can improve the reliability of its predictive models and ensure better decision-making for pricing and procurement strategies.

References

[1] Arjun Ruparelia (2023). Navigating the future with business forecasting. https://agicap.com/en/article/business-forecasting/

[2] Facil (2024). Facil Fastening Future Mobility
https://www.facil.be/en-gb/expertise/e-commerce/

[3] John W. Tukey (1962). ”Exploratory Data Analysis”
https://web.stanford.edu/class/archive/cs/cs448b/cs448b.1166/images/4/44/Lec448B-20160406.pdf

[4] lookerstudio
https://lookerstudio.google.com/u/0/reporting/b1d72206-768c-4d2b-a2dc-48880b843ff5/page/rac8D

[5] Fred ECONOMIC DATA (2024). Federal Reserve Bank
https://fred.stlouisfed.org/series/POILBREUSDM

[6] How to Deal with Missing Data
https://www.mastersindatascience.org/learning/how-to-deal-with-missing-data/

[7] Introduction to Machine Learning” by Alpaydin
https://dl.matlabyar.com/siavash/ML/Book/Ethem

[8] Hyperparameter tuning for machine learning models.
https://www.jeremyjordan.me/hyperparameter-tuning/

12 Appendix

12.1 Figures

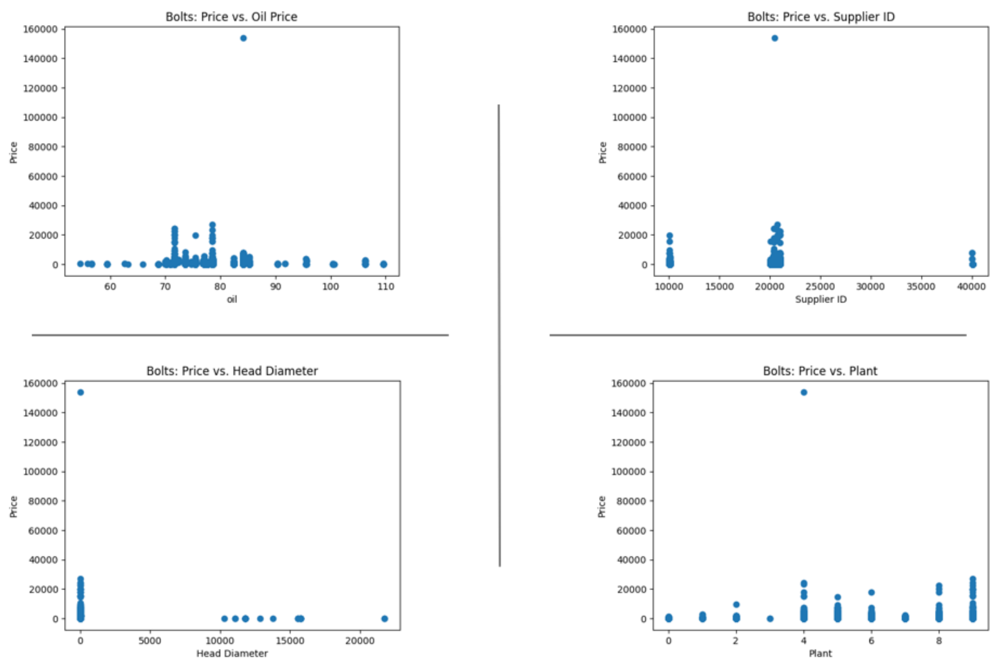

Figure 15: Data Visualization

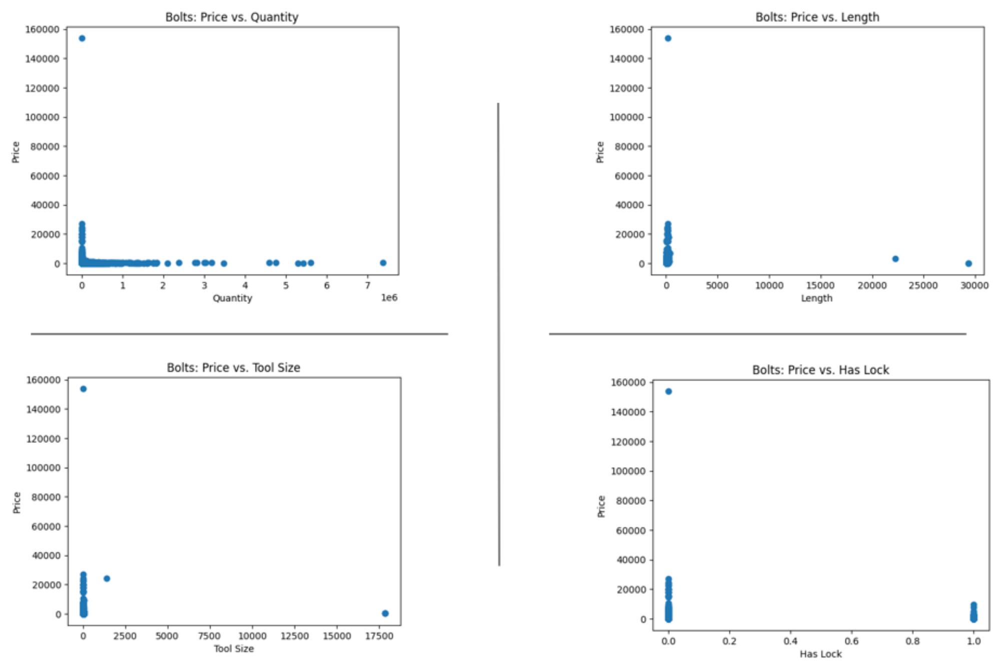

Figure 16: Data Visualization

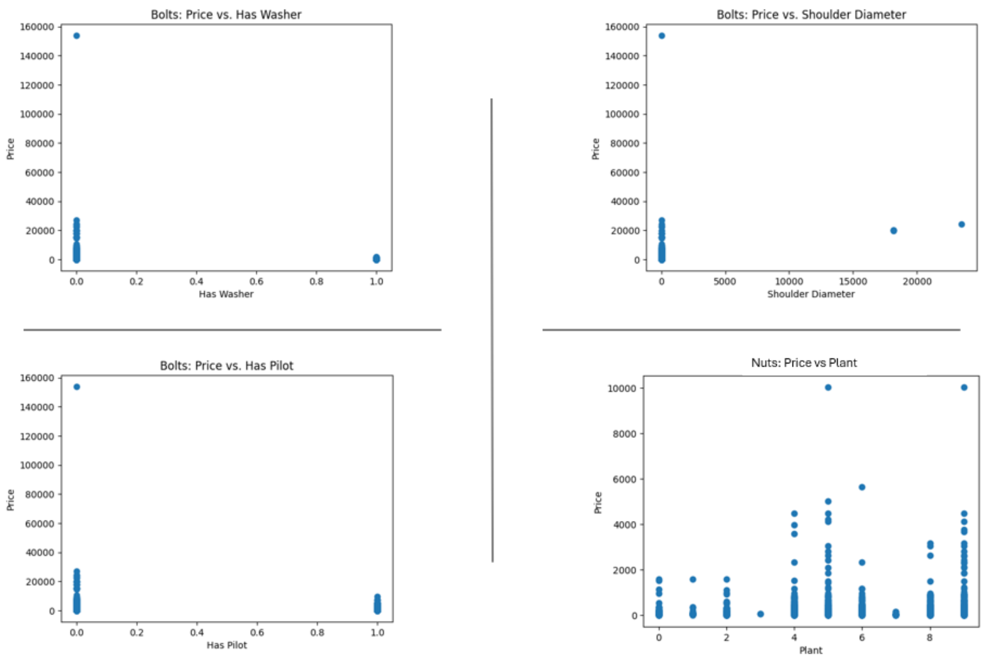

Figure 17: Data Visualization

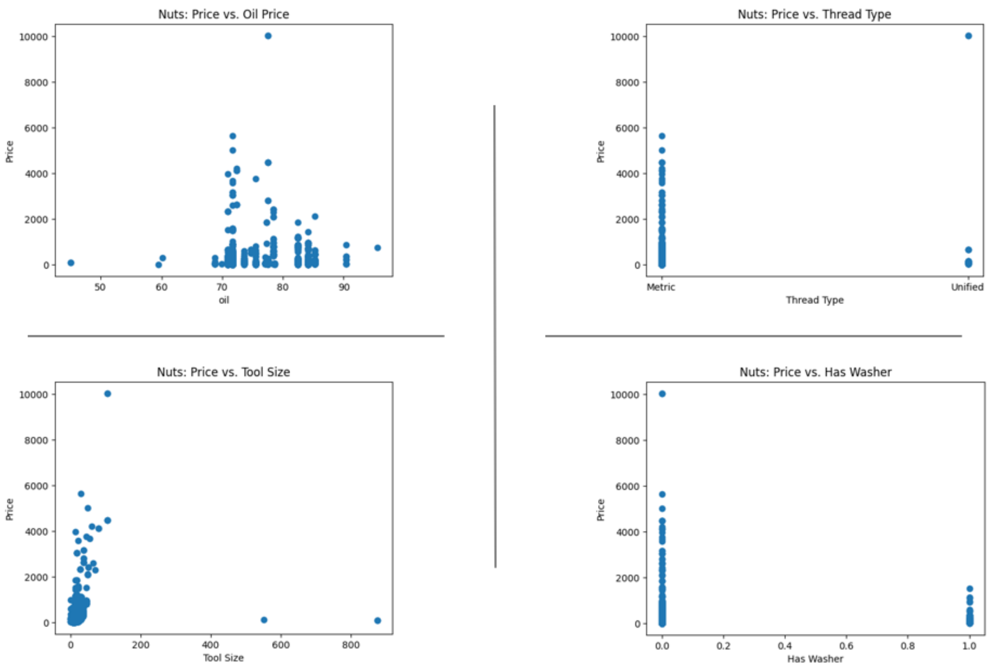

Figure 18: Data Visualization

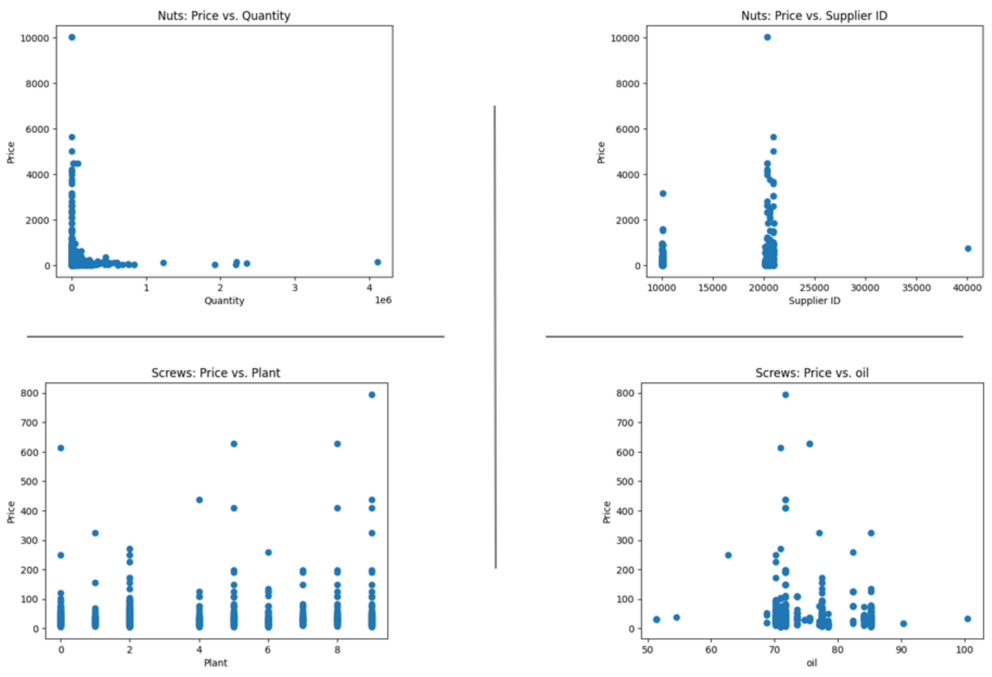

Figure 19: Data Visualization

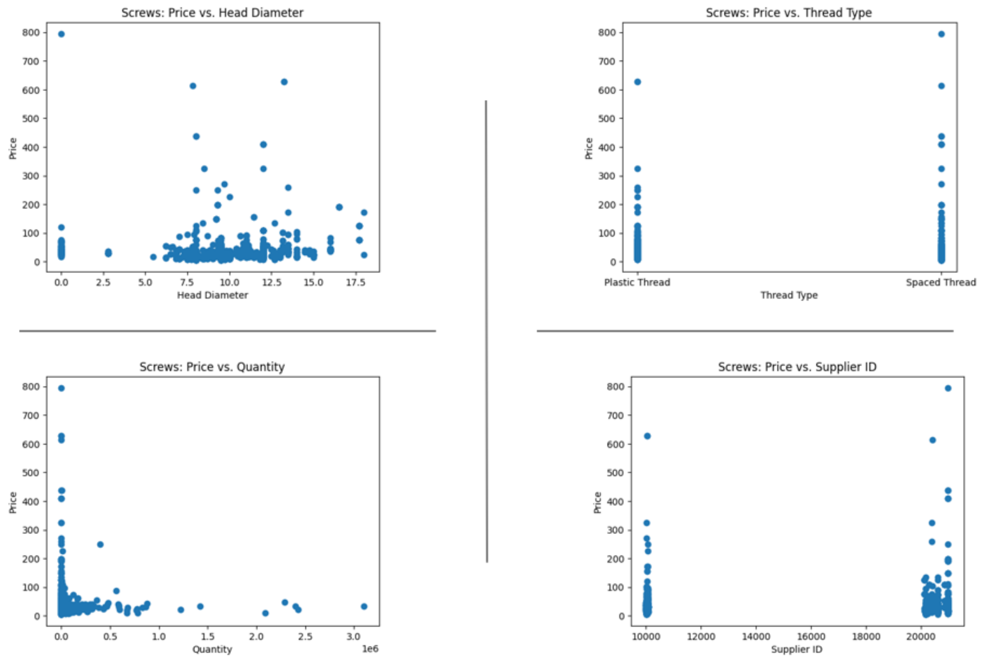

Figure 20: Data Visualization

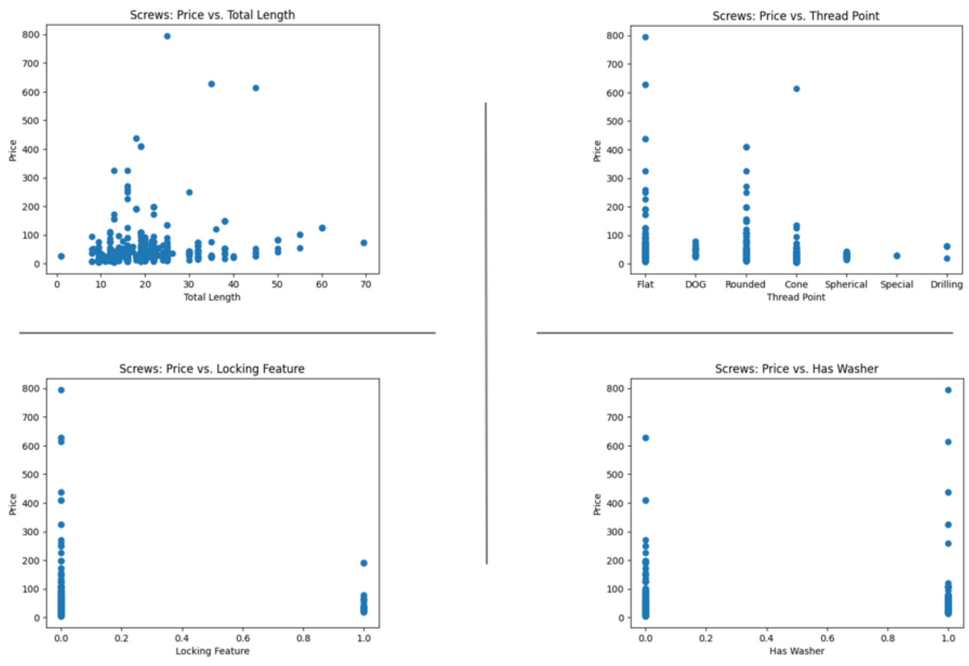

Figure 21: Data Visualization

12.2 Tables

COLUMN TITLE	DESCRIPTION
DISNAME	Program used as ex. InforXpert, SAP
XPPSPN	Part Number (PN)
FPID	Each PN has a unique ID no. in the program/system
CLIENTCOATING	Specifies the type of coating
FFINISH	Part Number ID number
COATINGID	A numeric code associated with each coating type
CATEGORY	Bolts/ Nuts/ Screws

Table 17: Description of excel facfasxppsparts

12.3 Code

Listing 1: Python Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_score
import numpy as np
!pip install matplotlib
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns
from sklearn.metrics import mean_squared_error
import re
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.ensemble import GradientBoostingRegressor

from google.colab import files
nuts_data = pd.read_excel('Nuts Final.xlsx')
screws_data = pd.read_excel('Screws Final.xlsx')
bolts_data = pd.read_excel('Bolts Final.xlsx')

# Display initial data information
print(nuts_data.head())
print(screws_data.head())
print(bolts_data.head())

# Function to check missing data in each dataframe
def check_missing_data(df, name):
    missing_data = df.isnull().sum()
    print(f"Missing data in {name}:\n{missing_data}\n")
    print(f"Total missing values in {name}: {missing_data.sum()}\n")

# Check missing data in each file
check_missing_data(nuts_data, 'Nuts Data')
check_missing_data(screws_data, 'Screws Data')
check_missing_data(bolts_data, 'Bolts Data')

price_summary = screws_data['Price'].describe()
# Print the summary statistics
print(price_summary)

price_summary = nuts_data['Price'].describe()
# Print the summary statistics
print(price_summary)

price_summary = bolts_data['Price'].describe()
# Print the summary statistics
print(price_summary)

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Plant'], nuts_data['Price'])
plt.xlabel('Plant')
plt.ylabel('Price')
plt.title('Price vs. Plant')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['oil'], nuts_data['Price'])
plt.xlabel('oil')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Oil Price')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Thread Type'], nuts_data['Price'])
plt.xlabel('Thread Type')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Thread Type')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Tool Size'], nuts_data['Price'])
plt.xlabel('Tool Size')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Tool Size')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Has Washer'], nuts_data['Price'])
plt.xlabel('Has Washer')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Has Washer')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Supplier ID'], nuts_data['Price'])
plt.xlabel('Supplier ID')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Supplier ID')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(nuts_data['Quantity'], nuts_data['Price'])
plt.xlabel('Quantity')
plt.ylabel('Price')
plt.title('Nuts: Price vs. Quantity')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Plant'], screws_data['Price'])
plt.xlabel('Plant')
plt.ylabel('Price')
plt.title('Screws: Price vs. Plant')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['oil'], screws_data['Price'])
plt.xlabel('oil')
plt.ylabel('Price')
plt.title('Screws: Price vs. oil')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Head Diameter'], screws_data['Price'])
plt.xlabel('Head Diameter')
plt.ylabel('Price')
plt.title('Screws: Price vs. Head Diameter')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Supplier ID'], screws_data['Price'])
plt.xlabel('Supplier ID')
plt.ylabel('Price')
plt.title('Screws: Price vs. Supplier ID')
plt.show()
```

```
# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Quantity'], screws_data['Price'])
plt.xlabel('Quantity')
plt.ylabel('Price')
plt.title('Screws: Price vs. Quantity')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Thread Type'], screws_data['Price'])
plt.xlabel('Thread Type')
plt.ylabel('Price')
plt.title('Screws: Price vs. Thread Type')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Head'], screws_data['Price'])
plt.xlabel('Head')
plt.ylabel('Price')
plt.title('Screws: Price vs. Head')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Total Length'], screws_data['Price'])
plt.xlabel('Total Length')
plt.ylabel('Price')
plt.title('Screws: Price vs. Total Length')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Locking Feature'], screws_data['Price'])
plt.xlabel('Locking Feature')
plt.ylabel('Price')
plt.title('Screws: Price vs. Locking Feature')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Thread Point'], screws_data['Price'])
plt.xlabel('Thread Point')
plt.ylabel('Price')
plt.title('Screws: Price vs. Thread Point')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(screws_data['Has Washer'], screws_data['Price'])
plt.xlabel('Has Washer')
plt.ylabel('Price')
plt.title('Screws: Price vs. Has Washer')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Plant'], bolts_data['Price'])
plt.xlabel('Plant')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Plant')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['oil'], bolts_data['Price'])
plt.xlabel('oil')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Oil Price')
plt.show()
```

```
# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Head Diameter'], bolts_data['Price'])
plt.xlabel('Head Diameter')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Head Diameter')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Supplier ID'], bolts_data['Price'])
plt.xlabel('Supplier ID')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Supplier ID')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Quantity'], bolts_data['Price'])
plt.xlabel('Quantity')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Quantity')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Head'], bolts_data['Price'])
plt.xlabel('Head')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Head')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Length'], bolts_data['Price'])
plt.xlabel('Length')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Length')
plt.show()



# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Tool Size'], bolts_data['Price'])
plt.xlabel('Tool Size')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Tool Size')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Has Lock'], bolts_data['Price'])
plt.xlabel('Has Lock')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Has Lock')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Has Pilot'], bolts_data['Price'])
plt.xlabel('Has Pilot')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Has Pilot')
plt.show()

# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Shoulder Diameter'], bolts_data['Price'])
plt.xlabel('Shoulder Diameter')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Shoulder Diameter')
plt.show()
```

```
# Scatter plot: Price vs. another factor (e.g., 'Factor1')
plt.scatter(bolts_data['Has Washer'], bolts_data['Price'])
plt.xlabel('Has Washer')
plt.ylabel('Price')
plt.title('Bolts: Price vs. Has Washer')
plt.show()

def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Assuming nuts_data, screws_data, and bolts_data are your DataFrames
nuts_data_clean = remove_outliers(nuts_data, 'Price')
screws_data_clean = remove_outliers(screws_data, 'Price')
bolts_data_clean = remove_outliers(bolts_data, 'Price')

# Display the cleaned data
print(nuts_data_clean)
print(screws_data_clean)
print(bolts_data_clean)

price_summary_without_outliers = screws_data_clean['Price'].describe()
# Print the summary statistics
print(price_summary_without_outliers)

price_summary_without_outliers = nuts_data_clean['Price'].describe()
# Print the summary statistics
```

```
print(price_summary_without_outliers)

price_summary_without_outliers = bolts_data_clean['Price'].describe()
# Print the summary statistics
print(price_summary_without_outliers)



# Load data
screws_data = pd.read_excel('Screws Final.xlsx')

# Remove outliers
screws_data_clean = remove_outliers(screws_data, 'Price')

# Technical features to consider
technical_features = ['Thread Type', 'Thread Size',
'Head','Head Diameter', 'Head Height','Thread Special',
'Tool Size', 'Tool', 'Total Length', 'Locking Feature',
'Locking Type','Thread Point', 'Has Washer', 'Washer',
'Washer Diameter', 'Washer Thickness', 'Weight']

# Prepare the data
X = screws_data_clean[technical_features]
y = screws_data_clean['Price']

# Identify non-numeric columns
non_numeric_columns = X.select_dtypes(include=['object']).columns

# Apply one-hot encoding
X = pd.get_dummies(X, columns=non_numeric_columns)

# Handle missing values
imputer = SimpleImputer(strategy='mean')
```

```
X = imputer.fit_transform(X)

# Get the feature names after one-hot encoding
feature_names = pd.get_dummies(screws_data_clean[technical_features],
columns=non_numeric_columns).columns

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train the model
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Get feature importance
feature_importance = model.feature_importances_

# Create a DataFrame for feature importance
importance_df = pd.DataFrame({'Feature': feature_names,

'Importance': feature_importance})

# Group by original feature names and sum the importances
importance_df['Original Feature']=

importance_df['Feature'].apply(lambda x: x.split('_')[0])
grouped_importance_df =
importance_df.groupby('Original Feature').sum().sort_values
(by='Importance',
ascending=False).head(10)

# Plot feature importance
```

```
plt.figure(figsize=(10, 6))
plt.barh(grouped_importance_df.index,

grouped_importance_df['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 10 Technical Features for Screws')
plt.gca().invert_yaxis()
plt.show()

# Load data
nuts_data = pd.read_excel('Nuts Final.xlsx')

# Remove outliers
nuts_data_clean = remove_outliers(nuts_data, 'Price')

# Technical features to consider
technical_features = ['Thread Type', 'Thread Size',
'Thread Pitch', 'Type', 'Diameter', 'Height',
'Property Class', 'Tool Size','Locking Feature',
'Locking Type', 'Has Paint Clearing', 'Has Washer',
'Washer', 'Washer Diameter', 'Washer Thickness', 'Weight']

# Prepare the data
X = nuts_data_clean[technical_features]
y = nuts_data_clean['Price']

# Identify non-numeric columns
non_numeric_columns = X.select_dtypes(include=['object']).columns

# Apply one-hot encoding
X = pd.get_dummies(X, columns=non_numeric_columns)
```

```
# Handle missing values
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Get the feature names after one-hot encoding
feature_names = pd.get_dummies(nuts_data_clean[technical_features],
columns=non_numeric_columns).columns

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train the model
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Get feature importance
feature_importance = model.feature_importances_

# Create a DataFrame for feature importance
importance_df = pd.DataFrame({'Feature': feature_names,
'Importance': feature_importance})

# Group by original feature names and sum the importances
importance_df['Original Feature'] =
importance_df['Feature'].apply(lambda x: x.split('_')[0])
grouped_importance_df = importance_df.groupby('Original
Feature').sum().sort_values(by='Importance',
ascending=False).head(10)

# Plot feature importance
```

```
plt.figure(figsize=(10, 6))
plt.barh(grouped_importance_df.index,
grouped_importance_df['Importance'],
color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 10 Technical Features for Nuts')
plt.gca().invert_yaxis()
plt.show()

# Load data
bolts_data = pd.read_excel('Bolts Final.xlsx')

# Remove outliers
bolts_data_clean = remove_outliers(bolts_data, 'Price')

# Technical features to consider
technical_features = ['Thread Type', 'Thread Size',
'Thread Pitch','Head', 'Head Diameter', 'Head Height',
'Length', 'Property Class', 'Thread Length', 'Thread Special',
'Tool Size', 'Tool', 'Total Length', 'Has Lock',
'Locking Type', 'Has Paint Clearing', 'Has Pilot',
'Pilot', 'Has Shoulder', 'Shoulder Diameter',
'Shoulder Height', 'TF', 'Has Washer','Washer',
'Washer Diameter', 'Washer Height', 'Has Wax', 'Weight']

# Prepare the data
X = bolts_data_clean[technical_features]
y = bolts_data_clean['Price']

# Identify non-numeric columns
non_numeric_columns = X.select_dtypes(include=['object']).columns
```

```
# Apply one-hot encoding
X = pd.get_dummies(X, columns=non_numeric_columns)

# Handle missing values
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Get the feature names after one-hot encoding
feature_names =
pd.get_dummies(bolts_data_clean[technical_features],
columns=non_numeric_columns).columns

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train the model
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Get feature importance
feature_importance = model.feature_importances_

# Create a DataFrame for feature importance
importance_df = pd.DataFrame({'Feature': feature_names,
'Importance': feature_importance})

# Group by original feature names and sum the importances
importance_df['Original Feature'] =
importance_df['Feature'].apply(lambda x: x.split('_')[0])
grouped_importance_df = importance_df.groupby('Original
Feature').sum().sort_values(by='Importance',
ascending=False).head(10)

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(grouped_importance_df.index,
grouped_importance_df['Importance'],
color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 10 Technical Features for Bolts')
plt.gca().invert_yaxis()
plt.show()

# Function to plot correlation heatmap
def plot_correlation_heatmap(data, title):
    numeric_features = data.select_dtypes(include=
    ['number']).column
    correlation_matrix = data[numeric_features].corr()
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix[['Price']], annot=True,
    cmap='coolwarm', vmin=-1, vmax=1)
    plt.title(title)
    plt.show()

# Plot correlation heatmap for each dataset
plot_correlation_heatmap(nuts_data_clean,
'Correlation with Price for Nuts')
plot_correlation_heatmap(screws_data_clean,
'Correlation with Price for Screws')
plot_correlation_heatmap(bolts_data_clean,
'Correlation with Price for Bolts')
```

```
# Clustering using K-means
from sklearn.cluster import KMeans

print("Bolts columns:", X_train_bolts.columns)
print("Nuts columns:", X_train_nuts.columns)
print("Screws columns:", X_train_screws.columns)

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder

# Function to select encoded feature columns
def select_encoded_features(feature_prefixes, all_features):
    selected_features = []
    for prefix in feature_prefixes:
        selected_features.extend([col for col in all_features
        if col.startswith(prefix)])
    return selected_features

# List of original features you want to select
bolts_features = ['Total Length', 'Length', 'Weight',
                  'Thread Length','Tool Size',
                  'Thread Pitch', 'Head Height',
                  'Thread Size', 'Property Class',
                  'Head Diameter']

nuts_features = ['Weight', 'Tool Size', 'Property Class',
                 'Height', 'Diameter',
                 'Washer', 'Thread Pitch',
                 'Thread Size', 'Washer Thickness',
                 'Type']

screws_features = ['Weight', 'Total Length', 'Head Height',
                    'Head Diameter', 'Tool',
                   'Head', 'Thread Point',
                   'Thread Special', 'Tool Size',
                   'Thread Size']

# Select relevant features for each category using the function
X_train_bolts = X_train_bolts[select_encoded_features(bolts_features,
X_train_bolts.columns)]
X_test_bolts = X_test_bolts[select_encoded_features(bolts_features,
X_test_bolts.columns)]
X_train_nuts = X_train_nuts[select_encoded_features(nuts_features,
X_train_nuts.columns)]
X_test_nuts = X_test_nuts[select_encoded_features(nuts_features,
X_test_nuts.columns)]
X_train_screws = X_train_screws[select_encoded_features(screws_features,
X_train_screws.columns)]
X_test_screws = X_test_screws[select_encoded_features(screws_features,
X_test_screws.columns)]

#KNN model training

# Train KNN models for each category
def train_knn(X_train, y_train, X_test, y_test):
    best_rmse = float('inf')
    best_k = None
    for k in range(1, 21):  # Try K values from 1 to 20
        knn = KNeighborsRegressor(n_neighbors=k)
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        rmse = mean_squared_error(y_test, y_pred, squared=False)
        if rmse < best_rmse:
            best_rmse = rmse
            best_k = k
    return best_k

# Pass the relevant train and test sets when calling the function
best_k_bolts = train_knn(X_train_bolts, y_train_bolts,
X_test_bolts, y_test_bolts)
best_k_nuts = train_knn(X_train_nuts, y_train_nuts,
X_test_nuts, y_test_nuts)
best_k_screws = train_knn(X_train_screws, y_train_screws,
X_test_screws, y_test_screws)

print(f"Best K for bolts: {best_k_bolts}")
print(f"Best K for nuts: {best_k_nuts}")
print(f"Best K for screws: {best_k_screws}")

from sklearn.metrics import mean_squared_error

# Train KNN models with the best K and calculate RMSE for each category

# Bolts
knn_bolts = KNeighborsRegressor(n_neighbors=best_k_bolts)
knn_bolts.fit(X_train_bolts, y_train_bolts)
y_pred_bolts = knn_bolts.predict(X_test_bolts)
rmse_bolts = mean_squared_error(y_test_bolts, y_pred_bolts,
squared=False)

# Nuts
knn_nuts = KNeighborsRegressor(n_neighbors=best_k_nuts)
```

```
knn_nuts.fit(X_train_nuts, y_train_nuts)
y_pred_nuts = knn_nuts.predict(X_test_nuts)
rmse_nuts = mean_squared_error(y_test_nuts, y_pred_nuts,
squared=False)

# Screws
knn_screws = KNeighborsRegressor(n_neighbors=best_k_screws)
knn_screws.fit(X_train_screws, y_train_screws)
y_pred_screws = knn_screws.predict(X_test_screws)
rmse_screws = mean_squared_error(y_test_screws, y_pred_screws,
squared=False)

print(f"RMSE for bolts: {rmse_bolts:.2f}")
print(f"RMSE for nuts: {rmse_nuts:.2f}")
print(f"RMSE for screws: {rmse_screws:.2f}")

import seaborn as sns
from sklearn.metrics import mean_squared_error

# Assuming you have the best_k for each category
knn_bolts = KNeighborsRegressor(n_neighbors=best_k_bolts)
knn_nuts = KNeighborsRegressor(n_neighbors=best_k_nuts)
knn_screws = KNeighborsRegressor(n_neighbors=best_k_screws)

# Fit the models
knn_bolts.fit(X_train_bolts, y_train_bolts)
knn_nuts.fit(X_train_nuts, y_train_nuts)
knn_screws.fit(X_train_screws, y_train_screws)

# Make predictions
y_pred_bolts = knn_bolts.predict(X_test_bolts)
y_pred_nuts = knn_nuts.predict(X_test_nuts)
```

```
y_pred_screws = knn_screws.predict(X_test_screws)

# Create DataFrames to compare actual vs predicted
df_bolts = pd.DataFrame({'Actual': y_test_bolts,
'Predicted': y_pred_bolts})
df_nuts = pd.DataFrame({'Actual': y_test_nuts,
'Predicted': y_pred_nuts})
df_screws = pd.DataFrame({'Actual': y_test_screws,
'Predicted': y_pred_screws})

# Plotting the violin plot
plt.figure(figsize=(14, 8))

plt.subplot(1, 3, 1)
sns.violinplot(data=pd.melt(df_bolts), x='variable', y='value',
palette="muted")
plt.title(f'Bolts (K={best_k_bolts})')
plt.xlabel('')

plt.subplot(1, 3, 2)
sns.violinplot(data=pd.melt(df_nuts), x='variable', y='value',
palette="muted")
plt.title(f'Nuts (K={best_k_nuts})')
plt.xlabel('')

plt.subplot(1, 3, 3)
sns.violinplot(data=pd.melt(df_screws), x='variable', y='value',
palette="muted")
plt.title(f'Screws (K={best_k_screws})')
plt.xlabel('')

plt.suptitle('Violin Plot of Actual vs Predicted Prices')
plt.show()

plt.figure(figsize=(14, 8))

plt.subplot(1, 3, 1)
plt.scatter(df_bolts['Actual'], df_bolts['Predicted'], alpha=0.5)
plt.plot([df_bolts['Actual'].min(), df_bolts['Actual'].max()],
         [df_bolts['Actual'].min(), df_bolts['Actual'].max()],
         color='red', lw=2)
plt.title(f'Bolts (K={best_k_bolts})')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 2)
plt.scatter(df_nuts['Actual'], df_nuts['Predicted'], alpha=0.5)
plt.plot([df_nuts['Actual'].min(), df_nuts['Actual'].max()],
         [df_nuts['Actual'].min(), df_nuts['Actual'].max()],
         color='red', lw=2)
plt.title(f'Nuts (K={best_k_nuts})')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 3)
plt.scatter(df_screws['Actual'], df_screws['Predicted'], alpha=0.5)
plt.plot([df_screws['Actual'].min(), df_screws['Actual'].max()],
         [df_screws['Actual'].min(), df_screws['Actual'].max()],
         color='red', lw=2)
plt.title(f'Screws (K={best_k_screws})')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.suptitle('Scatter Plot of Actual vs Predicted Prices')
plt.tight_layout()
plt.show()

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import mean_squared_error

# Define the parameter grid for hyperparameter tuning
param_grid_rf = {
    'n_estimators': [100, 200, 500],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Function to perform hyperparameter tuning
def tune_random_forest(X_train, y_train):
    rf = RandomForestRegressor(random_state=42)
    grid_search = GridSearchCV(estimator=rf,
    param_grid=param_grid_rf, scoring='neg_mean_squared_error',
    cv=5, n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_
    return best_model, grid_search.best_params_,
    -grid_search.best_score_

# Function to perform cross-validation
def cross_validate_model(model, X_train, y_train):
    cv_scores = cross_val_score(model, X_train, y_train,
    scoring='neg_mean_squared_error', cv=5)
    return -cv_scores.mean(), cv_scores.std()
```

```
# Bolts
best_rf_bolts, best_params_bolts, best_score_bolts =
tune_random_forest(X_train_bolts, y_train_bolts)
cv_mean_bolts, cv_std_bolts = cross_validate_model(best_rf_bolts,
X_train_bolts, y_train_bolts)
y_pred_bolts_rf = best_rf_bolts.predict(X_test_bolts)
rmse_bolts_rf = mean_squared_error(y_test_bolts, y_pred_bolts_rf,
squared=False)

# Nuts
best_rf_nuts, best_params_nuts, best_score_nuts =
tune_random_forest(X_train_nuts, y_train_nuts)
cv_mean_nuts, cv_std_nuts = cross_validate_model(best_rf_nuts,
X_train_nuts, y_train_nuts)
y_pred_nuts_rf = best_rf_nuts.predict(X_test_nuts)
rmse_nuts_rf = mean_squared_error(y_test_nuts, y_pred_nuts_rf,
squared=False)

# Screws
best_rf_screws, best_params_screws, best_score_screws =
tune_random_forest(X_train_screws, y_train_screws)
cv_mean_screws, cv_std_screws = cross_validate_model(best_rf_screws,
X_train_screws, y_train_screws)
y_pred_screws_rf = best_rf_screws.predict(X_test_screws)
rmse_screws_rf = mean_squared_error(y_test_screws, y_pred_screws_rf,
squared=False)

# Output the results
print(f"Best parameters for bolts: {best_params_bolts}")
print(f"Cross-validation RMSE for bolts: {cv_mean_bolts:.2f}
{cv_std_bolts:.2f}")
print(f"Test RMSE for bolts: {rmse_bolts_rf:.2f}")
print(f"Best parameters for nuts: {best_params_nuts}")
print(f"Cross-validation RMSE for nuts: {cv_mean_nuts:.2f}
{cv_std_nuts:.2f}")
print(f"Test RMSE for nuts: {rmse_nuts_rf:.2f}")

print(f"Best parameters for screws: {best_params_screws}")
print(f"Cross-validation RMSE for screws: {cv_mean_screws:.2f}
{cv_std_screws:.2f}")
print(f"Test RMSE for screws: {rmse_screws_rf:.2f}")

import matplotlib.pyplot as plt
import pandas as pd

# Prepare DataFrames for plotting
df_bolts_rf = pd.DataFrame({
    'Actual': y_test_bolts,
    'Predicted': y_pred_bolts_rf
})

df_nuts_rf = pd.DataFrame({
    'Actual': y_test_nuts,
    'Predicted': y_pred_nuts_rf
})

df_screws_rf = pd.DataFrame({
    'Actual': y_test_screws,
    'Predicted': y_pred_screws_rf
})

# Plot
plt.figure(figsize=(14, 8))
plt.subplot(1, 3, 1)
plt.scatter(df_bolts_rf['Actual'], df_bolts_rf['Predicted'], alpha=0.5)
plt.plot([df_bolts_rf['Actual'].min(), df_bolts_rf['Actual'].max()],
         [df_bolts_rf['Actual'].min(), df_bolts_rf['Actual'].max()],
         color='red', lw=2)
plt.title(f'Bolts (RF)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 2)
plt.scatter(df_nuts_rf['Actual'], df_nuts_rf['Predicted'], alpha=0.5)
plt.plot([df_nuts_rf['Actual'].min(), df_nuts_rf['Actual'].max()],
         [df_nuts_rf['Actual'].min(), df_nuts_rf['Actual'].max()],
         color='red', lw=2)
plt.title(f'Nuts (RF)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 3)
plt.scatter(df_screws_rf['Actual'], df_screws_rf['Predicted'], alpha=0.5)
plt.plot([df_screws_rf['Actual'].min(), df_screws_rf['Actual'].max()],
         [df_screws_rf['Actual'].min(), df_screws_rf['Actual'].max()],
         color='red', lw=2)
plt.title(f'Screws (RF)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.suptitle('Scatter Plot of Actual vs Predicted Prices (Random Forest)'
plt.tight_layout()
plt.show()
```

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import mean_squared_error

# Define the parameter grid for hyperparameter tuning
param_grid_gbr = {
    'n_estimators': [100, 200, 500],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Function to perform hyperparameter tuning
def tune_gradient_boosting(X_train, y_train):
    gbr = GradientBoostingRegressor(random_state=42)
    grid_search = GridSearchCV(estimator=gbr, param_grid=param_grid_gbr,
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_
    return best_model, grid_search.best_params_,
    -grid_search.best_score_

# Function to perform cross-validation
def cross_validate_model(model, X_train, y_train):
    cv_scores = cross_val_score(model, X_train, y_train,
    scoring='neg_mean_squared_error', cv=5)
    return -cv_scores.mean(), cv_scores.std()

# Bolts
best_gbr_bolts, best_params_bolts, best_score_bolts =
tune_gradient_boosting(X_train_bolts, y_train_bolts)
cv_mean_bolts, cv_std_bolts = cross_validate_model(best_gbr_bolts,
X_train_bolts, y_train_bolts)
y_pred_bolts_gbr = best_gbr_bolts.predict(X_test_bolts)
rmse_bolts_gbr = mean_squared_error(y_test_bolts, y_pred_bolts_gbr,
squared=False)

# Nuts
best_gbr_nuts, best_params_nuts, best_score_nuts =
tune_gradient_boosting(X_train_nuts, y_train_nuts)
cv_mean_nuts, cv_std_nuts = cross_validate_model(best_gbr_nuts,
X_train_nuts, y_train_nuts)
y_pred_nuts_gbr = best_gbr_nuts.predict(X_test_nuts)
rmse_nuts_gbr = mean_squared_error(y_test_nuts, y_pred_nuts_gbr,
squared=False)

# Screws
best_gbr_screws, best_params_screws, best_score_screws =
tune_gradient_boosting(X_train_screws, y_train_screws)
cv_mean_screws, cv_std_screws = cross_validate_model(best_gbr_screws,
X_train_screws, y_train_screws)
y_pred_screws_gbr = best_gbr_screws.predict(X_test_screws)
rmse_screws_gbr = mean_squared_error(y_test_screws, y_pred_screws_gbr,
squared=False)

# Output the results
print(f"Best parameters for bolts: {best_params_bolts}")
print(f"Cross-validation RMSE for bolts: {cv_mean_bolts:.2f}
{cv_std_bolts:.2f}")
print(f"Test RMSE for bolts: {rmse_bolts_gbr:.2f}")

print(f"Best parameters for nuts: {best_params_nuts}")
print(f"Cross-validation RMSE for nuts: {cv_mean_nuts:.2f}
{cv_std_nuts:.2f}")
```

```
print(f"Test RMSE for nuts: {rmse_nuts_gbr:.2f}")

print(f"Best parameters for screws: {best_params_screws}")
print(f"Cross-validation RMSE for screws: {cv_mean_screws:.2f}
{cv_std_screws:.2f}")
print(f"Test RMSE for screws: {rmse_screws_gbr:.2f}")

import matplotlib.pyplot as plt
import pandas as pd

# Prepare DataFrames for plotting
df_bolts_gbr = pd.DataFrame({
    'Actual': y_test_bolts,
    'Predicted': y_pred_bolts_gbr
})

df_nuts_gbr = pd.DataFrame({
    'Actual': y_test_nuts,
    'Predicted': y_pred_nuts_gbr
})

df_screws_gbr = pd.DataFrame({
    'Actual': y_test_screws,
    'Predicted': y_pred_screws_gbr
})

# Plot
plt.figure(figsize=(14, 8))

plt.subplot(1, 3, 1)
plt.scatter(df_bolts_gbr['Actual'], df_bolts_gbr['Predicted'],
alpha=0.5)
```

```
plt.plot([df_bolts_gbr['Actual'].min(), df_bolts_gbr['Actual'].max()],
         [df_bolts_gbr['Actual'].min(), df_bolts_gbr['Actual'].max()],
         color='red', lw=2)
plt.title(f'Bolts (GBR)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 2)
plt.scatter(df_nuts_gbr['Actual'], df_nuts_gbr['Predicted'],
alpha=0.5)
plt.plot([df_nuts_gbr['Actual'].min(), df_nuts_gbr['Actual'].max()],
         [df_nuts_gbr['Actual'].min(), df_nuts_gbr['Actual'].max()],
         color='red', lw=2)
plt.title(f'Nuts (GBR)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.subplot(1, 3, 3)
plt.scatter(df_screws_gbr['Actual'], df_screws_gbr['Predicted'],
alpha=0.5)
plt.plot([df_screws_gbr['Actual'].min(), df_screws_gbr['Actual'].max()],
         [df_screws_gbr['Actual'].min(), df_screws_gbr['Actual'].max()],
         color='red', lw=2)
plt.title(f'Screws (GBR)')
plt.xlabel('Actual')
plt.ylabel('Predicted')

plt.suptitle('Scatter Plot of Actual vs Predicted Prices
(Gradient Boosting)')
plt.tight_layout()
plt.show()
```