Data Science

Semester project

Analyzing Car Trends & Price Prediction from Pak Wheels Dataset 2023

Group Members:

- Abrar Saleem 20k-0129
- Owais 20k-0201
- Ali Qureshi 20k-0149

Course Instructor:

- Dr Nouman Durrani

Introduction:

The automotive industry is currently undergoing rapid changes driven by shifting consumer preferences, technological advancements, and evolving market dynamics. Accurately predicting car prices has become increasingly vital for stakeholders to make well-informed decisions. This report delves into predictive analysis of car prices using the extensive Pak Wheels dataset, illuminating trends that impact vehicle pricing. Through a Python-based analysis of the Pak Wheels car listings dataset, the report employs machine learning models, including Linear Regression, Convolutional Neural Networks (CNN), K-Nearest Neighbors (KNN), and ensemble methods. The analysis encompasses crucial steps in the data science pipeline, such as data cleaning, exploratory data analysis (EDA), and model application.

Dataset Overview:

The analysis hinges on the Pak Wheels dataset, a robust compilation of variables spanning car specifications, conditions, and market trends. Essential features, including make, model, year, mileage, fuel type, and, notably, price, form the core components of this dataset. Figure 1 shows all the fields in the dataset, while Figure 2 shows the dimensions of the dataset.

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 89956 entries, 0 to 89955
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ad_url           89956 non-null  object
 1   title            89956 non-null  object
 2   location         89954 non-null  object
 3   model_year       87449 non-null  float64
 4   price            89954 non-null  object
 5   mileage          89954 non-null  object
 6   engine_type      89001 non-null  object
 7   transmission     89954 non-null  object
 8   registered_in    89954 non-null  object
 9   color            88093 non-null  object
 10  assembly         89956 non-null  object
 11  engine_capacity  89954 non-null  object
 12  body_type        81407 non-null  object
 13  ad_last_updated  89954 non-null  object
 14  car_features     83435 non-null  object
 15  description      89954 non-null  object
dtypes: float64(1), object(15)
memory usage: 11.7+ MB
```

Figure 1: All fields in the dataset

```
In [3]: df.shape

Out[3]: (89956, 16)
```

Figure 2: Shape of the dataset

A nuanced comprehension of the dataset is pivotal for deriving meaningful insights and constructing precise predictive models. Derived from 'pakwheels.csv,' the dataset scrutinized here encompasses a wealth of information on car listings, incorporating diverse attributes such as model year, mileage, engine type, transmission, and price. With a mix of numerical and categorical data, this dataset provides a fertile ground for predictive modeling. Initial exploration reveals potential challenges like duplicates, missing values, and variations in data formats. The primary objective of this analysis is to meticulously transform and prepare the dataset, unlocking valuable insights, and establishing resilient machine learning models that accurately predict car prices.

Data Cleaning:

The initial phase focuses on importing necessary libraries and examining the dataset. Key cleaning steps involve handling duplicates, addressing missing values, standardizing numerical fields, and leveraging natural language processing (NLP) for text cleaning. The goal is to prepare a clean and uniform dataset for subsequent analysis.

- **Handling Missing Values and Duplicate data:**

To ensure data integrity, missing values are identified using statistical methods. Imputation strategies, such as mean or regression-based techniques, are employed, and decisions are made regarding the removal of incomplete data.

```
In [6]: df.isna().sum()

Out[6]: ad_url              0
        title               0
        location            2
        model_year       2507
        price               2
        mileage             2
        engine_type       955
        transmission        2
        registered_in       2
        color            1863
        assembly            0
        engine_capacity     2
        body_type        8549
        ad_last_updated     2
        car_features     6521
        description         2
        dtype: int64
```

Figure 3: checking for null values in every field.

An example would be the extraction of the model year from the title field. Figure 4 shows the number of missing values before the extraction, then text analysis is performed on the title field and the model year is extracted from the tittle field, Figure 5 shows that only 19 values remain that do not have an assigned model year after this text analysis and as shown in figure 6 they are subsequently dropped.

```
In [25]: print('NAN',df['model_year'].isna().sum())

         NAN 2507
```

Figure 4: No of null values before any operation is performed.

```
In [28]: print('NAN',df['model_year'].isna().sum())

         NAN 19
```

Figure 5: No of null values after performing text analysis.

```
In [30]: df.dropna(subset=['model_year'], inplace=True)
         print('NAN',df['model_year'].isna().sum())

         NAN 0
```

Figure 6: Dropping the remaining null values.

Similar actions are performed on every field to drastically decrease the null values in the dataset. Figure 7 shows that almost every field has their null values treated in the dataset.

```
In [70]: df.isna().sum()

Out[70]: location              0
         model_year            0
         price                 0
         mileage               0
         engine_type           0
         transmission          0
         registered_in         0
         color                 0
         assembly              0
         engine_capacity       0
         body_type          8545
         ad_last_updated       0
         car_features       6518
         brand                 0
         vehicle               0
         make                  0
         ev                    0
         current_city          0
         has_ac                0
         cleaned_description   0
         age                   0
         dtype: int64
```

Figure 7: Final state of the null values.

- **Dropping Duplicate Data:**

To ensure the integrity and accuracy of the dataset, a critical step involves the identification and removal of duplicate entries. Utilizing panda's functionality, as shown in Figure 8, the script systematically identifies and drops duplicate rows, enhancing the dataset's cleanliness and eliminating redundant information. This process contributes to a more reliable foundation for subsequent analyses, preventing potential distortions in statistical metrics and model training.

```
In [5]: if df.duplicated().any():
            df = df.drop_duplicates()
```

Figure 8: Dropping the duplicate records in the dataset.

- **Converting String values to Correct Formats:**

In this phase, meticulous steps are taken to convert string representations to numeric formats, ensuring a standardized and homogeneous dataset. A notable example is the transformation of the 'price' column. Original values, expressed in strings like '10 lakhs,' undergo systematic

conversions to their numeric equivalents, such as '1000000.' Figures 9 and 10 show the values in the price field before and after the normalization.

```
In [9]: df.price.head()

Out[9]: 0    PKR 16.5 lacs
        1       PKR 7 lacs
        2     PKR 7.6 lacs
        3      PKR 70 lacs
        4       PKR 8 lacs
        Name: price, dtype: object
```

Figure 9:Values in price field before normalization.

```
        df['price']

Out[21]: 0        1650000.0
         1         700000.0
         2         760000.0
         3        7000000.0
         4         800000.0
                     ...
         89951    3000000.0
         89952    3750000.0
         89953    2235000.0
         89954    1550000.0
         89955     730000.0
         Name: price, Length: 89956, dtype: float64
```

Figure 10: Values in price field after normalization

Similar transformations are applied to other pertinent columns, promoting consistency and precision across the dataset. The date of the add uploaded was converted to date time from string as shown in figure 11.

```
In [7]: df['ad_last_updated'] = pd.to_datetime(df['ad_last_updated'])
```

This process not only enhances the uniformity of monetary representations but also facilitates more accurate analyses and machine learning model training.

- **Natural Language Processing (NLP):**

Incorporating Natural Language Processing techniques, the 'description' and the 'title' fields undergo a transformative journey. A specialized cleaning function rids the text of HTML tags and non-alphanumeric characters, fostering a cleaner and more manageable corpus. Subsequent steps involve tokenization and stop-word removal, facilitating the extraction of valuable information from the textual descriptions. This NLP preprocessing not only refines the 'cleaned description' column but also opens avenues for text-based analyses and enriches the dataset's overall predictive power. This helps us in feature engineering as we extract new fields from these existing fields. Figure 12 shows the cleaning of the description field.

```
def clean_text(text):
    text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
    text = re.sub(r'[^a-zA-Z0-9]', ' ', text)  # Remove special characters
    text = text.lower()  # Convert to lowercase
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text)
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)
```

```
df['cleaned_description'] = df['description'].apply(clean_text)
```

Figure 12: Natural Language processing of the description field.

- **Feature Engineering:**

In the pursuit of refining the dataset, an essential aspect is the creation of meaningful features derived from existing columns. New features are created, enhancing the predictive power of the model. Leveraging insights embedded in the 'description' field, a dedicated column named 'has ac' is introduced, explicitly indicating whether a car possesses air conditioning features. Figure 13 showcases how this was accomplished using text analysis.

```
ac_presence_patterns = r'\b(?:ac|air conditioning|air conditioning working|ac working|ac available|air conditioning available)\b'
ac_absence_patterns = r'\b(?:no ac|no air conditioning|ac not working|air conditioning not working|ac not available|air conditior

df['has_ac'] = np.where(df['car_features'].str.contains(ac_presence_patterns, case=False, na=False) & ~df['car_features'].str.cor

df['has_ac'].value_counts()

1    78092
0    11845
Name: has_ac, dtype: int64
```

Figure 13: Extraction of has_ac field from the description field.

This extraction process not only adds granularity to the dataset but also lays the groundwork for more nuanced analyses. Another crucial feature, 'ev,' is engineered based on the 'transmission' column. Vehicles with 'kWh' in their transmission details are designated as electric cars ('ev'), offering a distinctive attribute for further exploration. Figure 14 shows this extraction.

```
df['ev'] = df['engine_capacity'].str.contains('kWh').astype(int)

df['engine_capacity'].str.contains(' kWh').value_counts()

False    89803
True       134
Name: engine_capacity, dtype: int64
```

Figure 14: Creating the new ev field.

A new field called age was also extracted from model year field, shown in Figure 15.

```
In [67]: current_year = pd.to_datetime('today').year
         df['age'] = current_year - df['model_year']
```

Figure 15: Field age extracted from field model year.

Field current city was extracted from the field location to standardize the location category, Figure 16 shows how it was done.

```
city_pattern = r'(\b(?:Karachi|Lahore|Islamabad|Rawalpindi|Multan|Faisalabad|Quetta|Peshawar|Sia

# Extract city using regular expressions
df['current_city'] = df['location'].str.extract(city_pattern, expand=False, flags=re.IGNORECASE)
```

Figure 16: Creating a new field called current city.

These meticulous enhancements not only contribute to a more informative dataset but also set the stage for sophisticated predictive modeling.

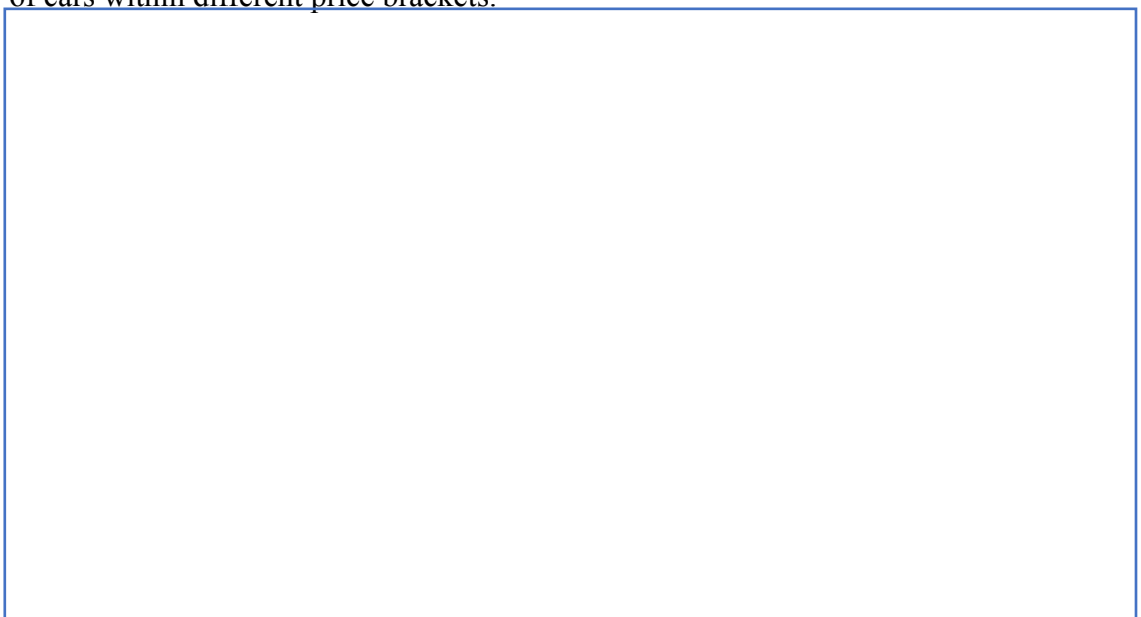
The data cleaning process played a pivotal role as a crucial precursor to meaningful exploratory data analysis. Key steps included addressing missing values, eliminating duplicates, and standardizing data formats to ensure dataset integrity. The transformation of textual information through Natural Language Processing (NLP) techniques enriched the dataset by extracting valuable features. Feature engineering further enhanced the dataset's depth, introducing novel attributes for subsequent analyses. The resulting clean and refined dataset set the stage for more accurate insights, predictive modeling, and informed decision-making.

Exploratory Data Analysis:

This section employs visualizations to gain insights into the dataset. Embarking on the Exploratory Data Analysis (EDA) phase, our focus shifts from data preparation to uncovering insights and patterns within the refined dataset. This stage involves statistical and visual exploration to glean a deeper understanding of the car listings data. Through various charts, graphs, and statistical measures, we aim to unravel correlations, trends, and outliers. This exploratory journey lays the foundation for subsequent modeling, providing essential context and informing feature selection. Let's delve into the diverse facets of the dataset, unraveling its intricacies to extract meaningful observations and guide our analytical journey.

This visualization showcased in figure 17 shows the top 30 car brands based on the number of cars sold. The bar plot effectively communicates the distribution of sales among these brands, providing a quick and insightful overview of the market landscape.

This histogram in figure 18 illustrates the distribution of car prices under 1 crore in the dataset. The plot offers a clear representation of the price range, emphasizing the frequency of cars within different price brackets.



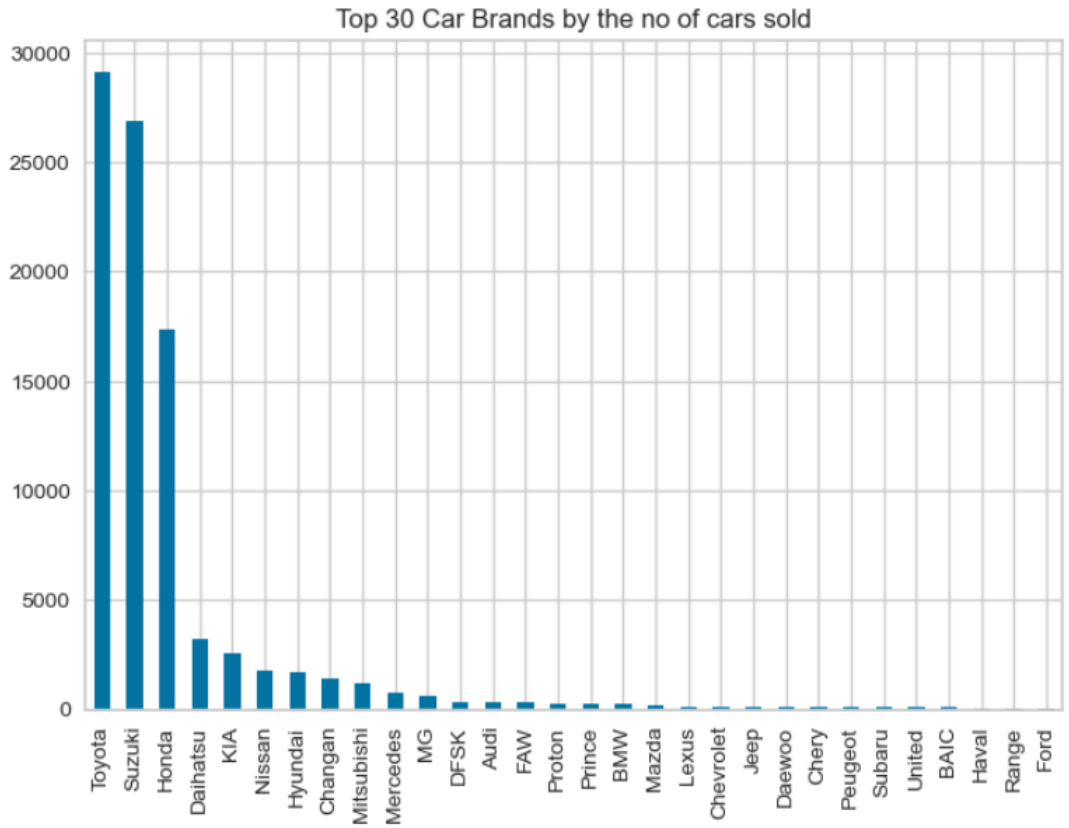


Figure 17: Top 30 Car Brands by the no. of cars sold.

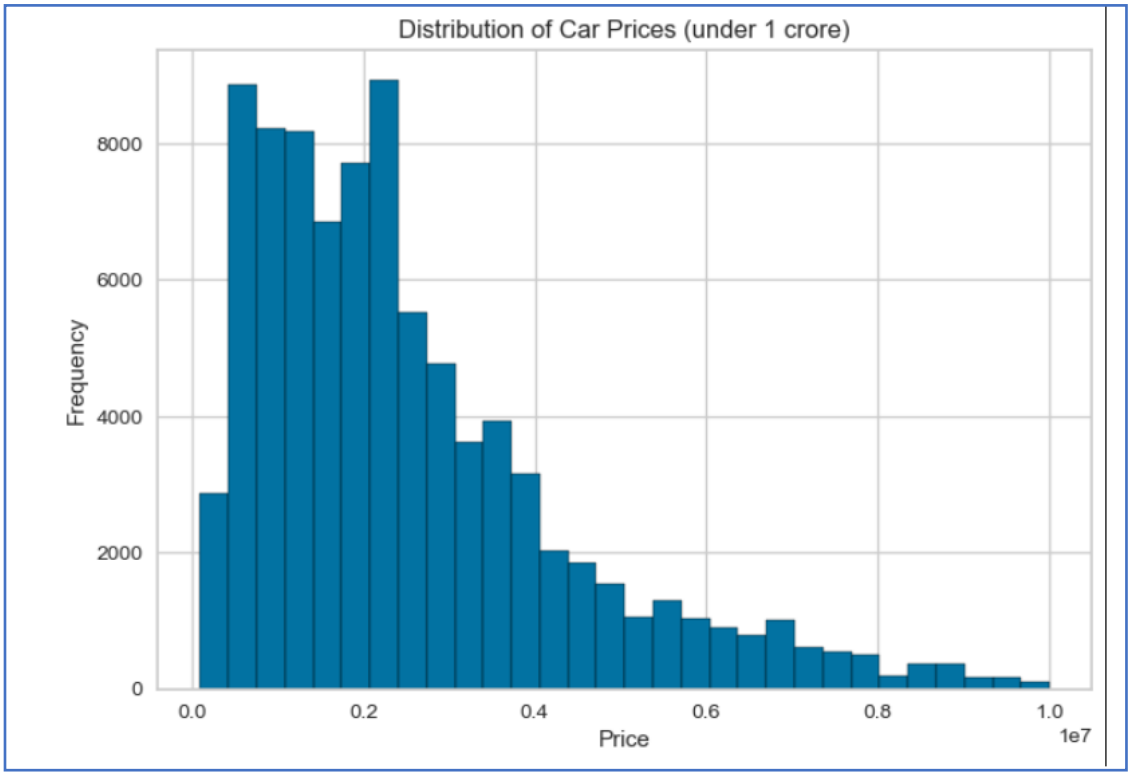


Figure 18: Distribution of car prices

The pie chart in figure 19 visualizes the distribution of cars based on their transmission types. It provides a concise overview of the proportion of cars with different transmission mechanisms, contributing to a comprehensive understanding of the dataset's composition.

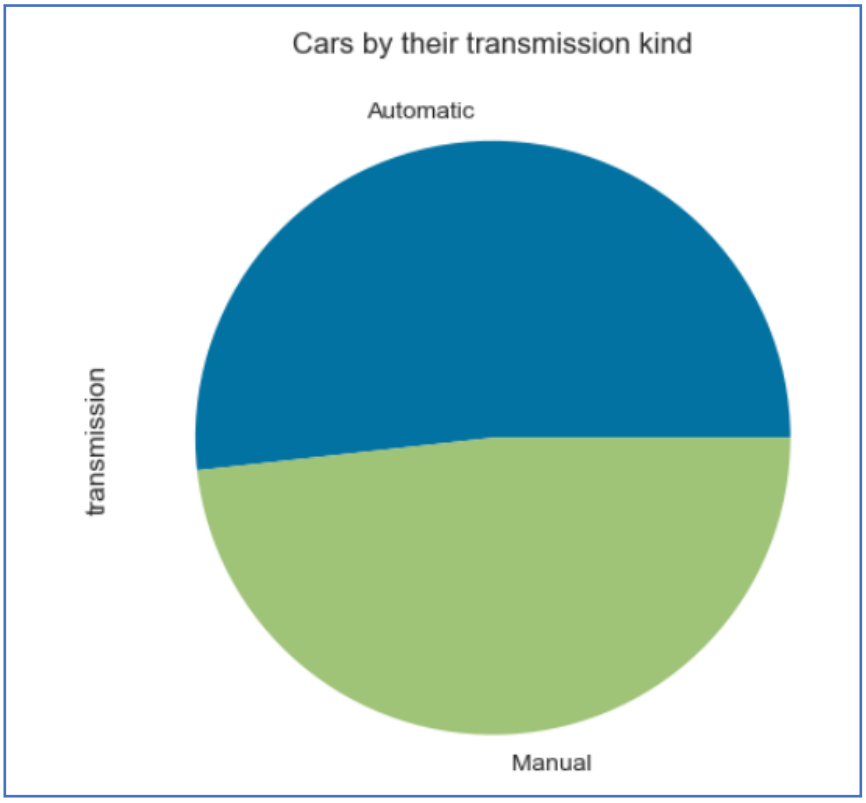
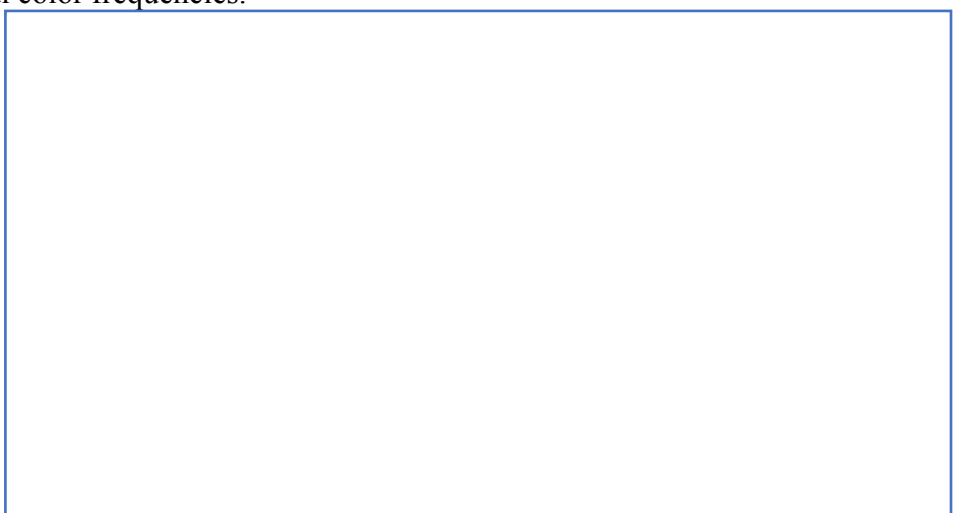


Figure 19: Cars by transmission.

The combined use of a pie chart and a bar chart in Figure2 20 and 21 presents a dual perspective on the distribution of cars by color. The pie chart offers a visual representation of the overall distribution, highlighting the relative proportions of each color category. Meanwhile, the accompanying bar chart provides a detailed count of cars for each color, offering a more granular insight into the dataset's color composition. This visual synthesis aids in both quick assessments of predominant colors and a more detailed exploration of individual color frequencies.



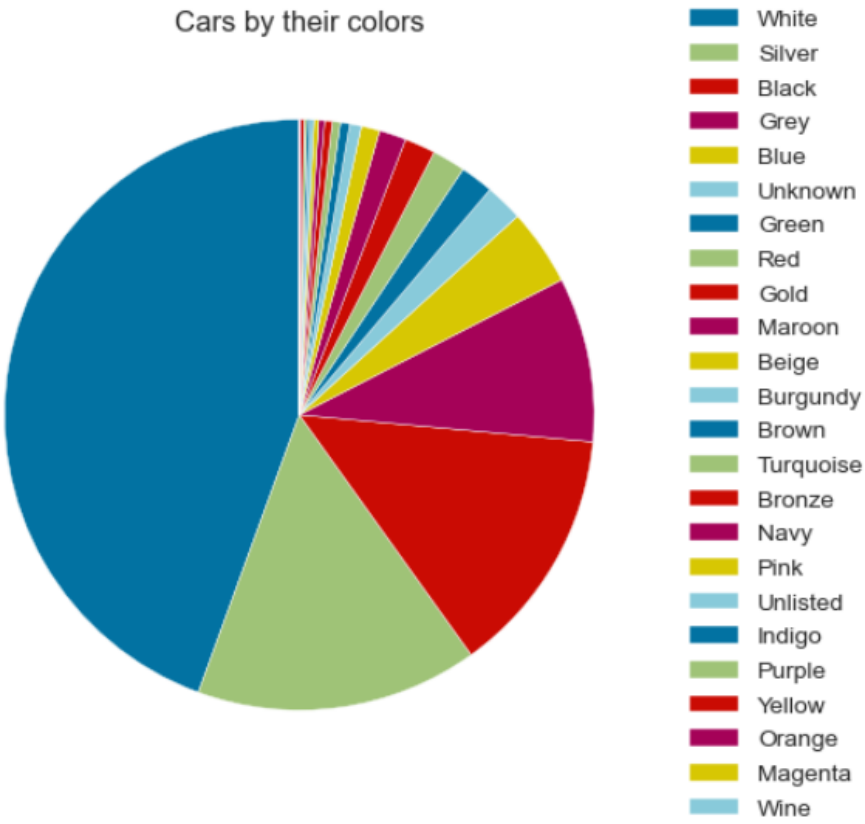


Figure 20: Cars by their colors in a pie char.

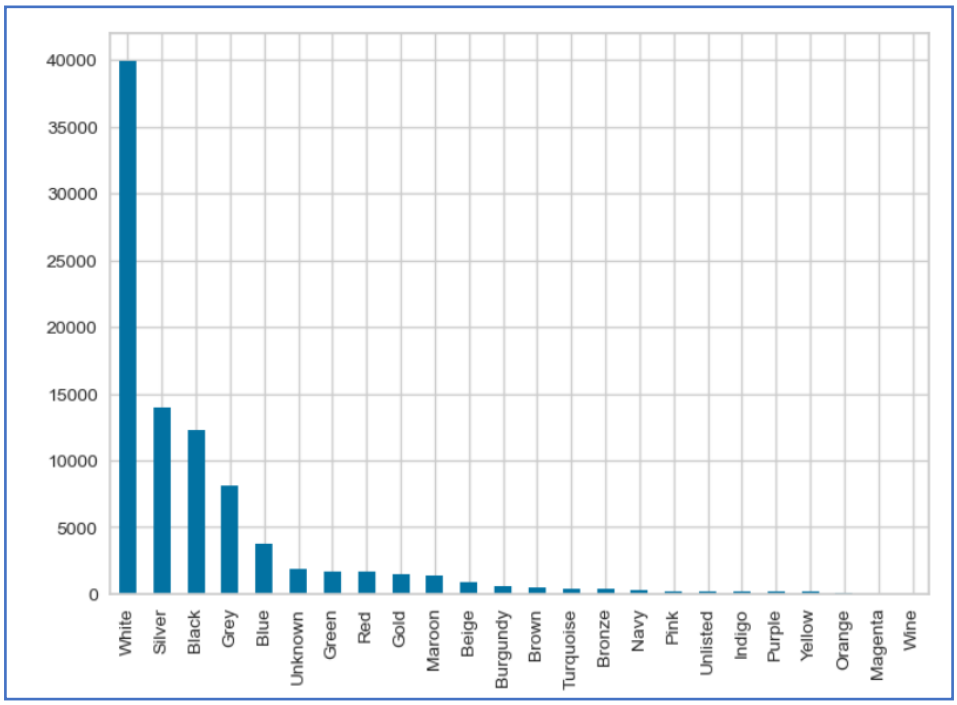


Figure 21: Cars by their colors in a bar graph

The correlation heatmap visually encapsulates the relationships between various numerical features within the dataset. Each cell's color intensity corresponds to the strength and direction of the correlation between two variables. Annotations provide precise correlation coefficients, aiding in the identification of significant patterns and potential multicollinearity. This heatmap serves as a comprehensive overview of the interplay between different attributes, offering valuable insights for subsequent analytical and modeling endeavors.

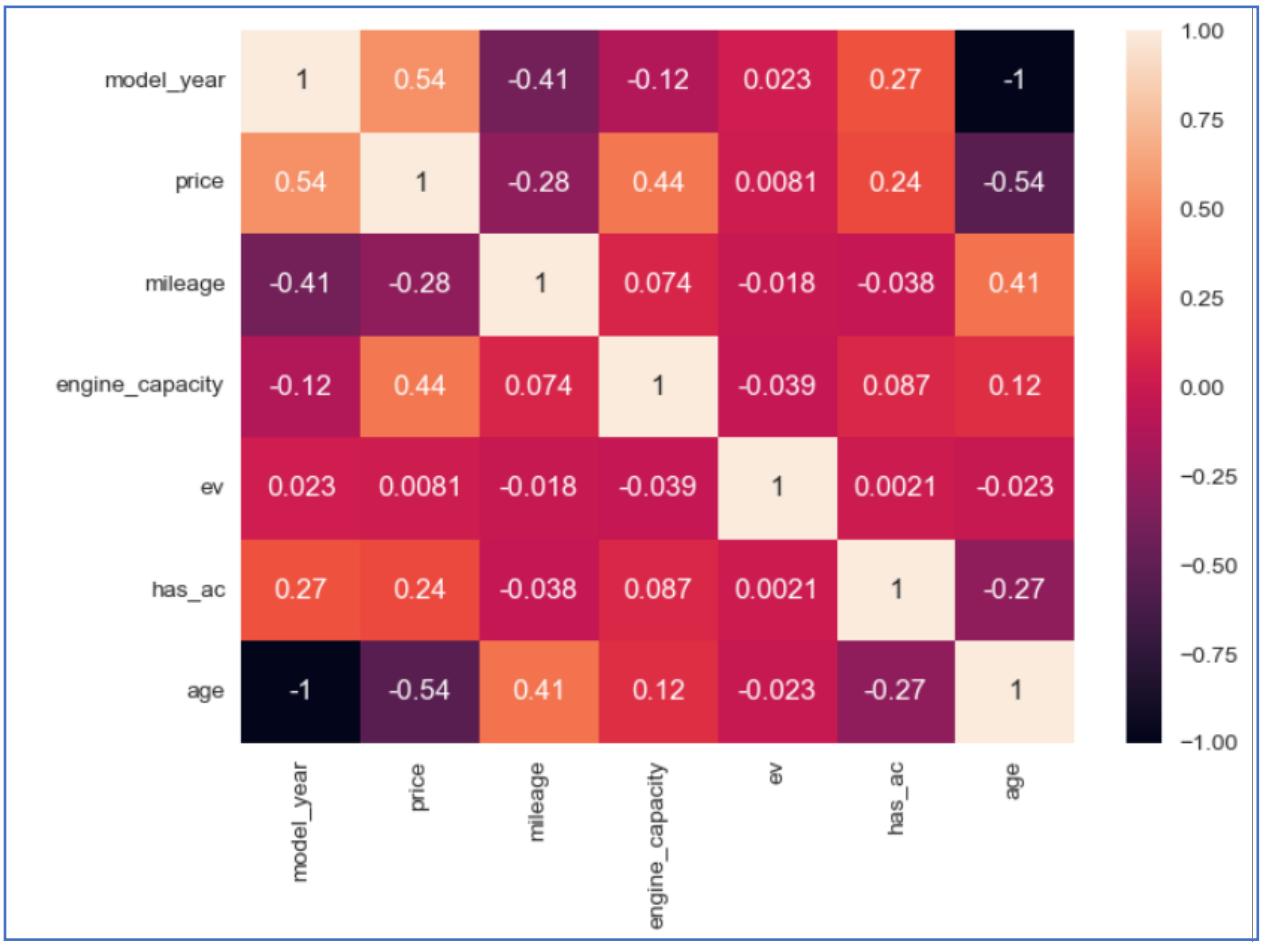


Figure 22: Correlation between the fields in the dataset

- **<u>Outlier Detection and Removal:</u>**

Outlier detection during Exploratory Data Analysis (EDA) plays a crucial role in understanding the distribution and patterns within a dataset. By employing statistical methods and visualization techniques, potential outliers—data points significantly deviating from the norm—are identified. Box plots, scatter plots, and histograms are instrumental in visually highlighting data points that fall outside the expected range. In our analysis, such outlier detection ensures a more accurate representation of the dataset, enabling us to make informed decisions about how to handle these extreme values. This step is pivotal for robust

statistical analyses and reliable machine learning model development, as it helps prevent skewed insights and biases that outliers might introduce.
First, we made a box plot as shown in figure 23 and saw that there were 3 main outliers, so we removed them.

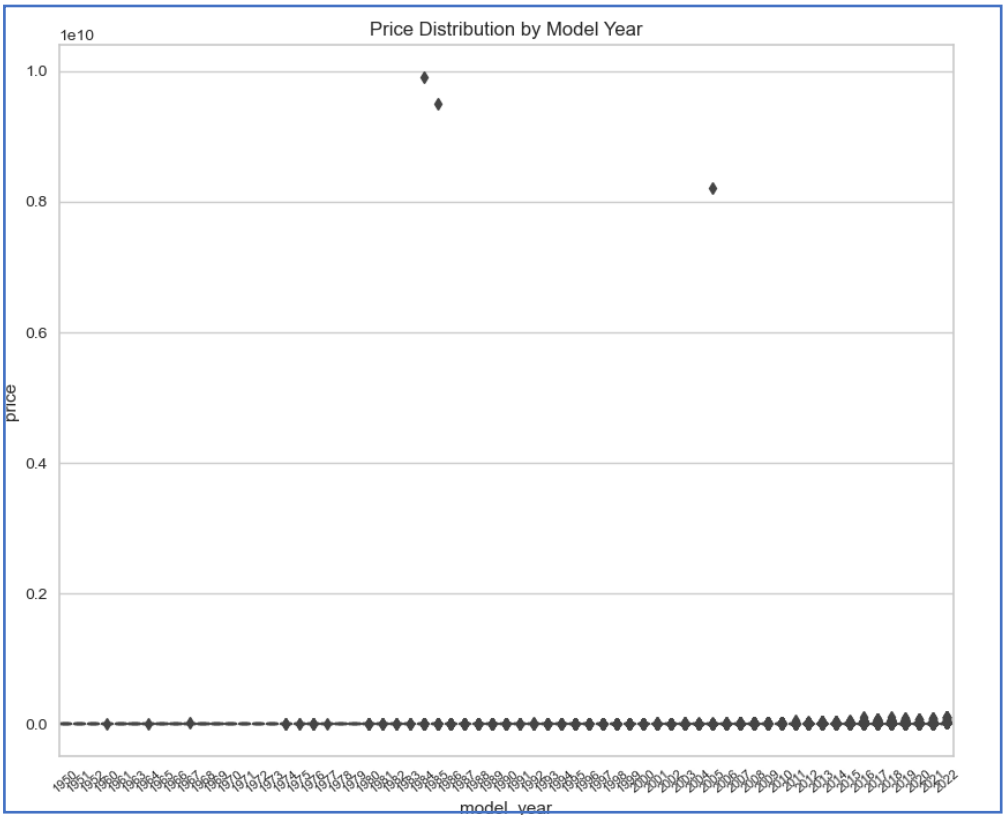


Figure 23: Box plot showcasing 3 main outliers.

Then, we made the box plot again as shown in figure 24 and saw that 90% of our data lies below 10000000-unit price so we removed them as well.
Figure 25 shows the box plot of the prices after we removed the 10% anomalous data.

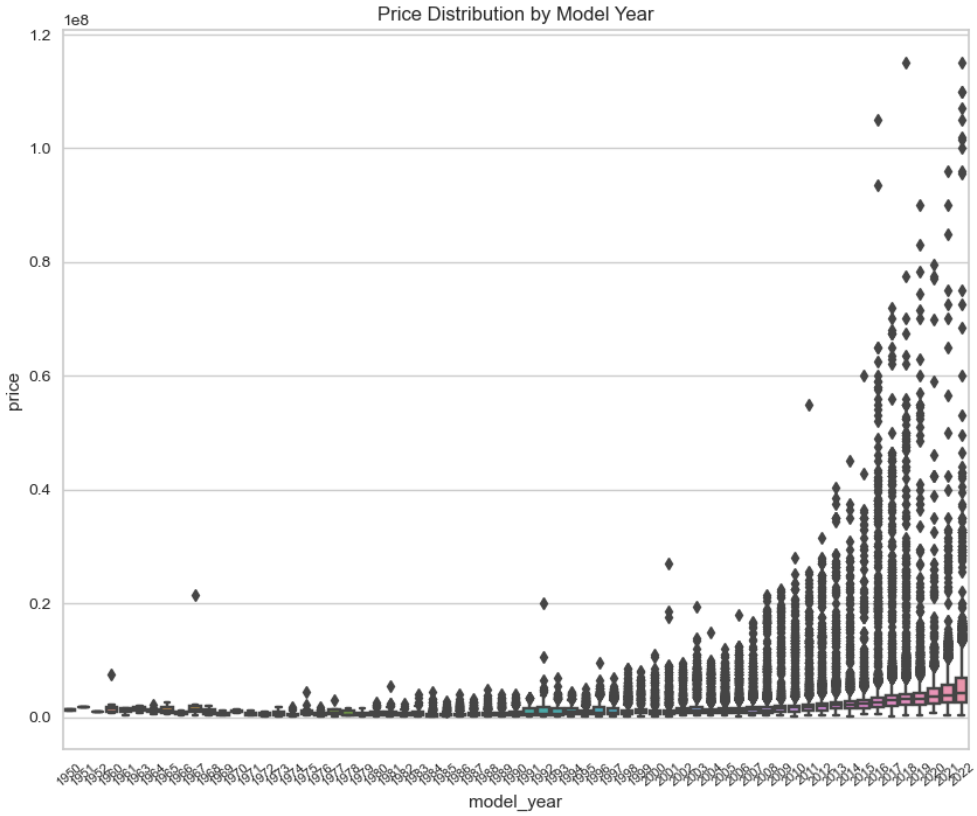


Figure 24: Box plot after removing the 3 outliers.

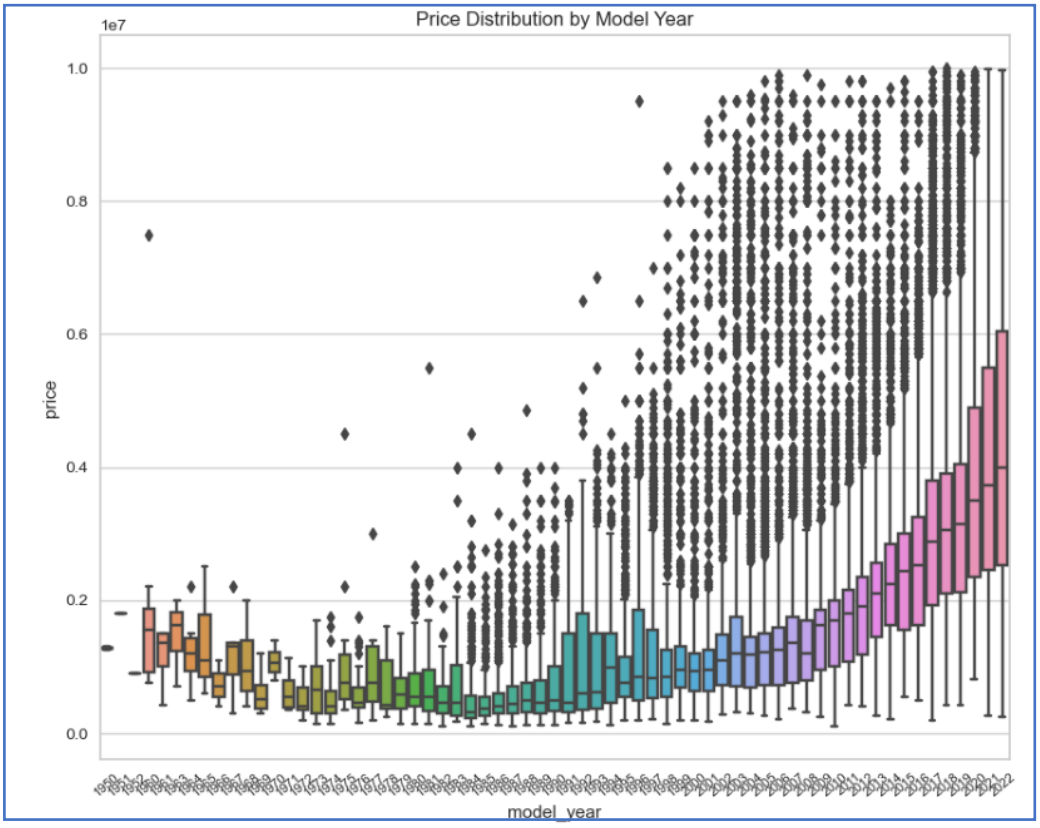


Figure 25: Boxplot after removing 10% outliers.

Model Training:

Model training is a pivotal phase in our analytical journey, where the prepared dataset takes center stage for developing predictive models. Leveraging machine learning algorithms, we split the dataset into training and testing sets to evaluate the model's performance accurately. A diverse array of algorithms, such as regression, decision trees, or ensemble methods, is employed based on the nature of the prediction task. Through iterative processes of training, validation, and fine-tuning, the model learns patterns and relationships within the data, optimizing its predictive capabilities. Rigorous evaluation metrics, including accuracy, precision, recall, and F1 score, guide the selection of the most effective model. This stage not only marks the culmination of meticulous data preparation but also sets the foundation for making informed predictions and deriving valuable insights from the dataset.

- **Linear Regression:**

Initially, a linear regression model was trained on the dataset, intending to capture linear relationships between input features and car prices. However, the results proved unsatisfactory, with a staggering Mean Squared Error (MSE) of 1.49 trillion. The model struggled to predict prices accurately, often yielding predictions nearly double the actual values. This outcome indicated that the complex relationships in the data required a more sophisticated approach, leading to the exploration of alternative models to achieve better predictive performance.

```
X = df[['model_year', 'mileage', 'engine_capacity','ev','has_ac']]
y = df['price']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 1486568247079.261
```

Figure 26: Training a LR model.

```
: new_data = pd.DataFrame({'model_year': [2007], 'mileage': [100000], 'engine_capacity': [1800], 'ev': [0], 'has_ac': [1]})

# Make predictions for the new data
predicted_price = model.predict(new_data)

rounded_price = round(predicted_price[0] / 1000) * 1000

# Display the rounded price
print(f'Predicted Rounded Price: {rounded_price}')
```

```
Predicted Rounded Price: 2605000
```

Figure 27: Prediction on LR model, expect value was 1.65e6 but got 2.6e6

- **<u>Convolutional Neural Networks:</u>**

Transitioning to Convolutional Neural Network (CNN) training provided an improvement over the initial linear regression attempt, yet the Mean Squared Error (MSE) persisted at a considerable level, measuring 741 billion. Recognizing the need for further refinement, the focus shifted to Mean Absolute Error (MAE) for prediction evaluation. Additionally, an exploration of alternative models was undertaken to ensure a comprehensive understanding of the dataset and to enhance prediction accuracy.

```
input_layers = []

for col in categorical_columns:
    input_dim = len(label_encoders[col].classes_)
    output_dim = min(5, input_dim)
    input_layer = Input(shape=(1,))
    embedding_layer = Embedding(input_dim=input_dim, output_dim=output_dim)(input_layer)
    input_layers.append(input_layer)
    embedding_layers.append(embedding_layer)

# Concatenate the embeddings
concatenated_embeddings = concatenate(embedding_layers, axis=-1)

# Adjusted kernel size to 1
output = Conv1D(filters=64, kernel_size=1, activation='relu')(concatenated_embeddings)
output = MaxPooling1D(pool_size=1)(output)  # Decreased pooling factor
output = Flatten()(output)
output = Dense(64, activation='relu')(output)
output = Dense(1, activation='linear')(output)

# Create a Model instance
model = Model(inputs=input_layers, outputs=output)

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Convert input data to numpy arrays
X_train_inputs = [X_train[col].to_numpy() for col in categorical_columns]
X_test_inputs = [X_test[col].to_numpy() for col in categorical_columns]

# Fit the model
model.fit(X_train_inputs, y_train, epochs=10, batch_size=32, validation_data=(X_test_inputs, y_test))
```

Figure 28: Code snippet from CNN training.

```
In [95]: # Split X_test into separate inputs
         X_test_inputs = [X_test[col].to_numpy() for col in categorical_columns]

         # Evaluate the model
         loss = model.evaluate(X_test_inputs, y_test)
         print(f'Mean Squared Error on Test Data: {loss}')

546/546 [==============================] - 0s 827us/step - loss: 741010112512.0000
Mean Squared Error on Test Data: 741010112512.0
```

Figure 29: CNN Testing and MSE

- **More models used:**

Exploring alternative avenues, the implementation of K-Means clustering, and K-Nearest Neighbors (KNN) models yielded results that fell short of expectations. For K-Means, an elbow graph suggested the optimal number of clusters to be four, yet the clustered data exhibited limited cohesion. Each cluster displayed a mix of diverse features, making it challenging to discern meaningful patterns. The KNN model, while showing improvement over previous attempts, still yielded a high Mean Squared Error (MSE) of approximately 763 billion, highlighting the complexity of the dataset. In response to these challenges, the focus shifted towards ensemble learning models, specifically adopting a bagging approach to harness the collective power of multiple models for enhanced predictive accuracy.

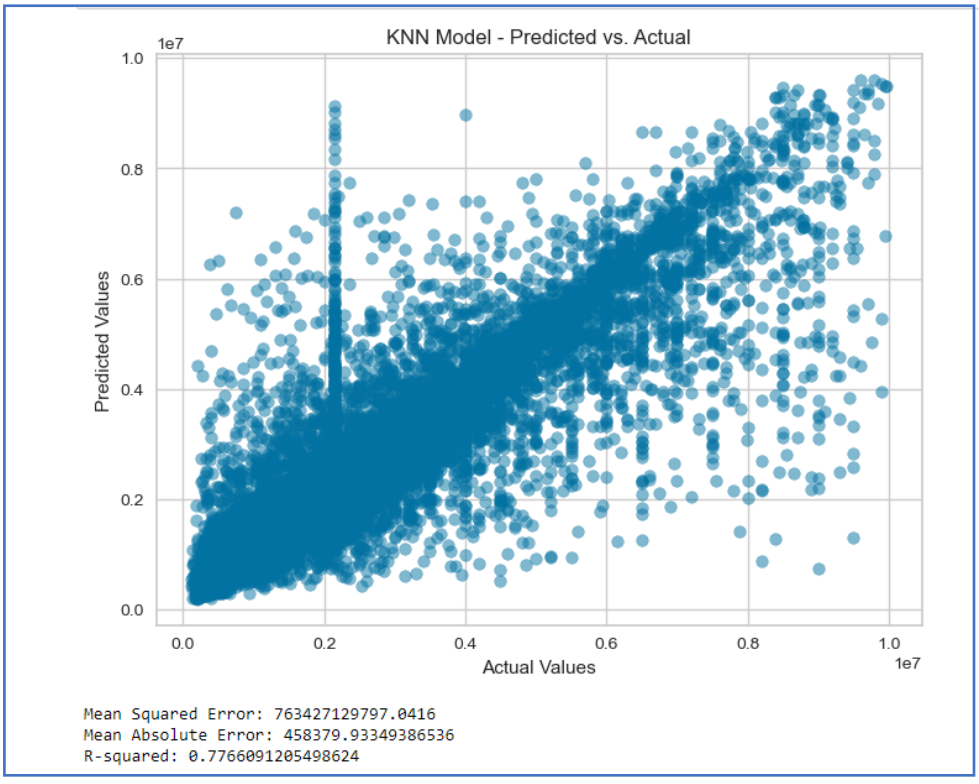
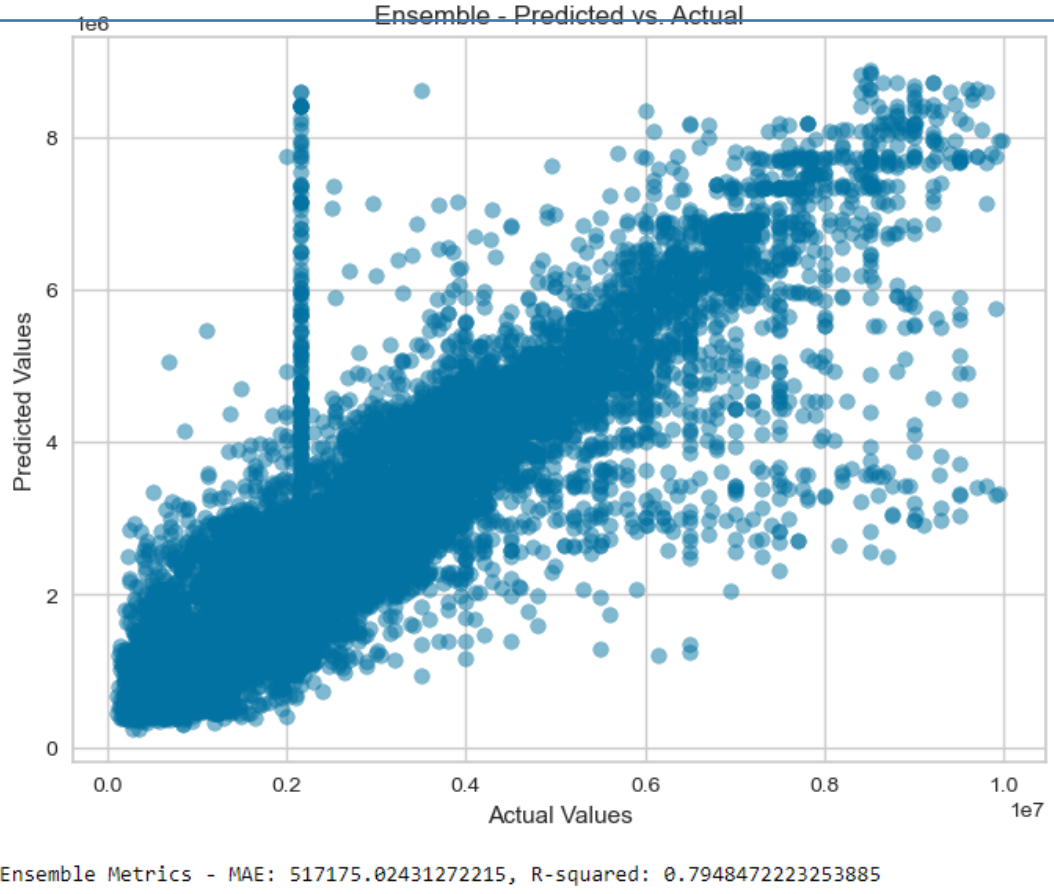


Figure 30: Results for KNN model

- **Ensemble Learning (CNN):**

In the pursuit of refining predictive performance, a bagging ensemble approach was employed on the Convolutional Neural Network (CNN) model, undergoing five consecutive training iterations. This concerted effort resulted in a notable improvement, as indicated by ensemble metrics: Mean Absolute Error (MAE) of 517,175 and an impressive R-squared value of 0.79. While this demonstrated a substantial leap forward from prior attempts, a final endeavor involved implementing a bagging ensemble approach with Random Forest, aiming to extract further predictive accuracy from the diverse set of base learners. This final stage encapsulates a comprehensive strategy to extract the utmost insight from the dataset, ensuring the robustness and reliability of the predictive model.



Ensemble Metrics - MAE: 517175.02431272215, R-squared: 0.7948472223253885

Figure 31: Results of ensembled CNN model.

- **Ensemble Learning (Random Forest):**

In the culmination of our model refinement journey, the implementation of a bagging ensemble approach with Random Forest yielded outstanding results. The ensemble metrics exhibited a remarkable Mean Absolute Error (MAE) of 251,970 and an exceptionally high R-squared value of 0.91. This outcome signifies a substantial enhancement in predictive accuracy compared to previous attempts, showcasing the efficacy of leveraging ensemble learning techniques to harness the collective strength of diverse models. The high R-squared value suggests that the ensemble model captures a significant portion of the variance in the target variable, providing a robust foundation for making accurate predictions on car prices in our dataset.

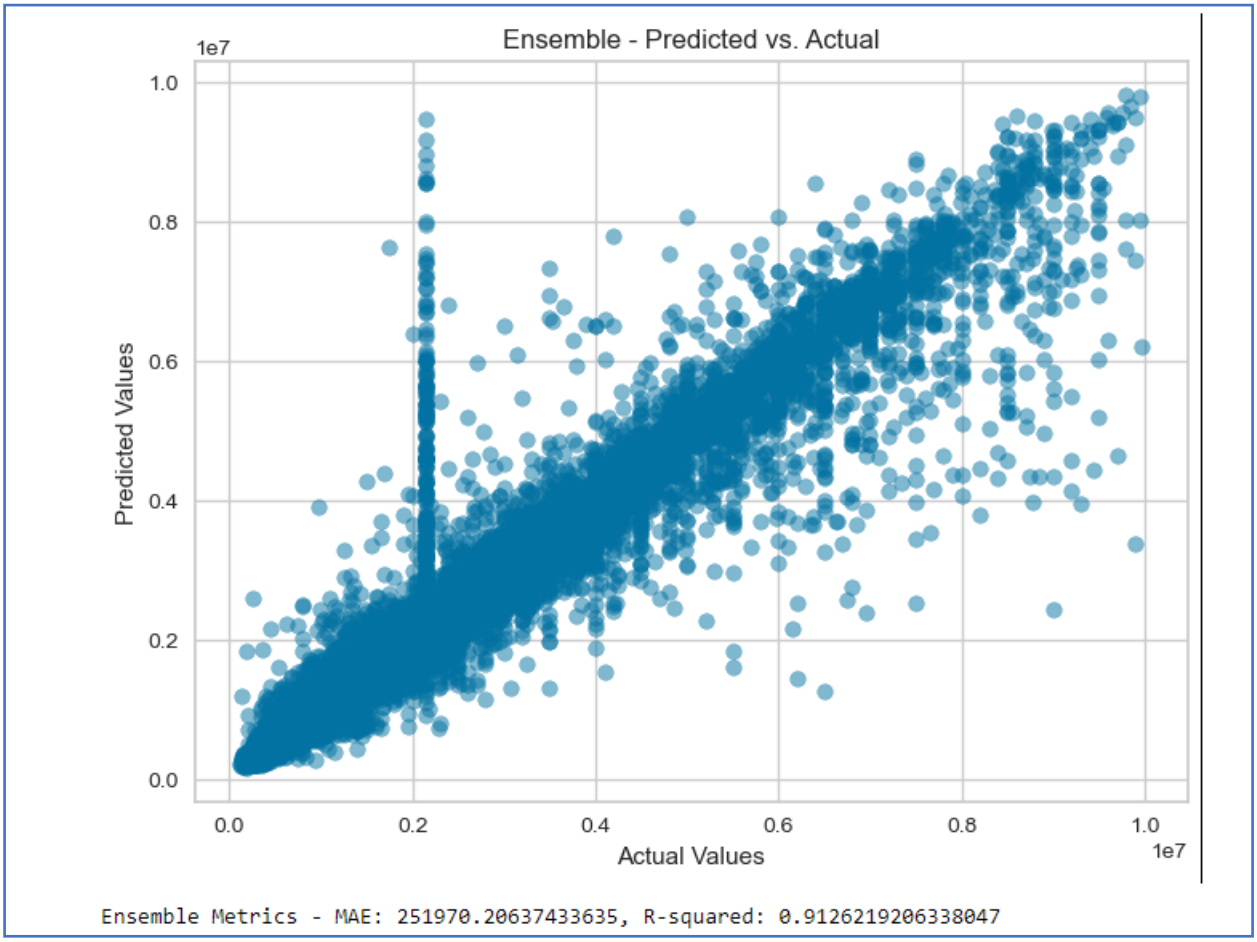


Figure 32: Results of ensemble Random Forest

In the realm of machine learning for predicting car prices, our iterative journey unfolded a spectrum of methodologies, each contributing to the evolution of our predictive models. Commencing with linear regression, despite encountering initial challenges with an alarming Mean Squared Error, we pivoted to convolutional neural networks (CNN), witnessing improved results yet realizing the imperative for further enhancement. Our exploration delved into K-means clustering and KNN models, shedding light on their limitations. With a

strategic shift towards ensemble learning, the bagging approach, initially applied to CNN, yielded significant strides in predictive accuracy. The zenith was reached with the implementation of a bagging ensemble on a Random Forest model, showcasing an impressive MAE of 251,970 and a robust R-squared value of 0.91. This not only attests to the intricacies of predicting car prices but also underscores the prowess of ensemble learning in amalgamating the strengths of diverse models for a more robust and accurate predictive outcome.

Output