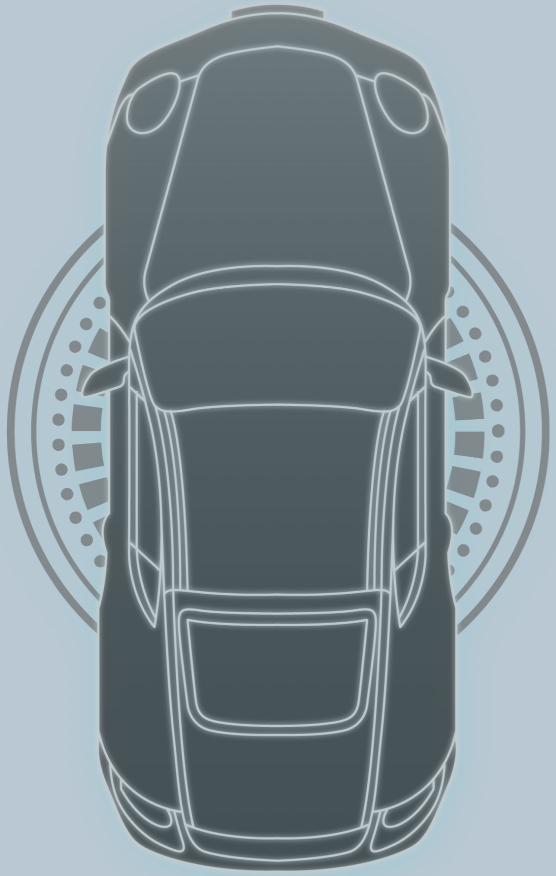
Predicting the City-Cycle Fuel Consumption in Miles per Gallon of a Car



WE LEAD - Team1

Team Members

Kyriaki Fanaridou

Aristea Kotoula

Sofia Mamali

Alexandra Matzari

Despoina Voulgari

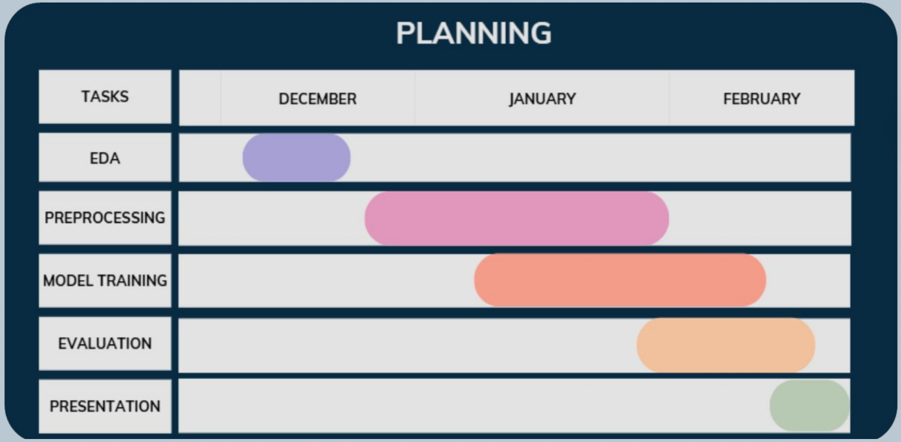
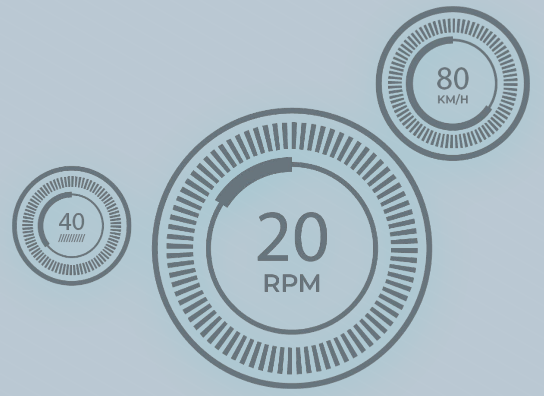


TABLE OF CONTENTS

01

Introduction

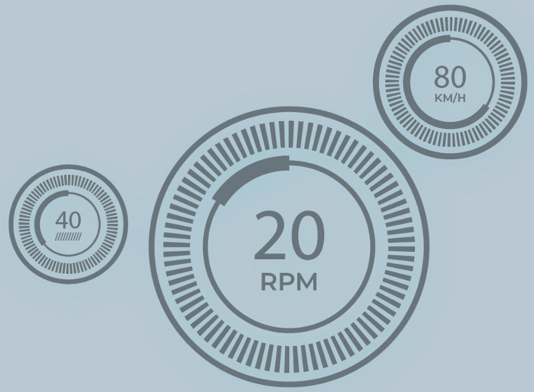


Introduction

- **Objective**: Develop a Proof of Concept (POC) to predict car fuel efficiency (MPG) for car rental companies.
- **Data**: Utilize the Auto MPG Dataset (398 instances, 8 attributes) by transforming it from a regression to a classification problem using percentiles.
- **Task**: Explore, preprocess, build, train, and evaluate a machine learning model using scikit-learn to classify MPG (low, medium, high).
- **Goal**: Provide insights for car rental companies to optimize fleet updates based on predicted fuel efficiency.

02

EDA & Preprocessing



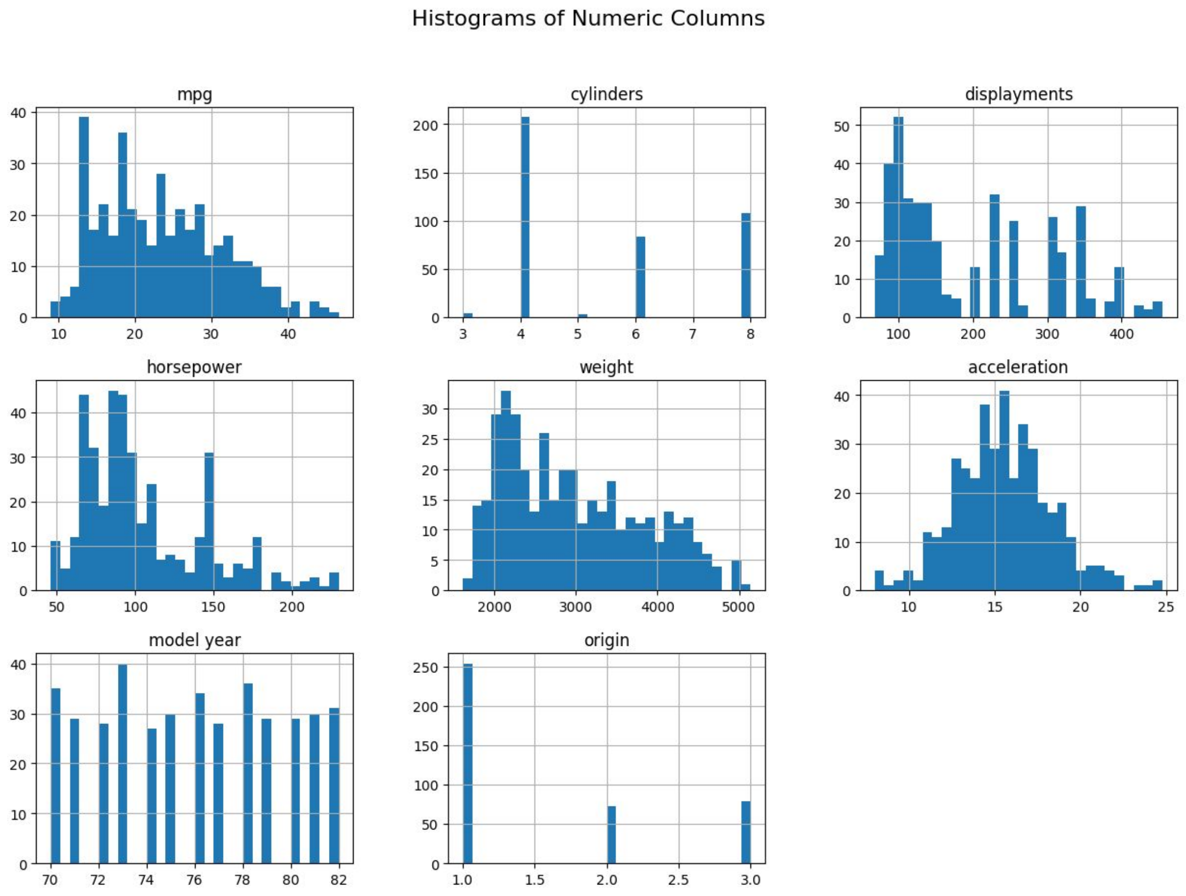
EDA

- Null values were found, no duplicates found
- 4 columns were dropped
- Missing Data of mpg, horsepower were dropped

Missing Values

mpg	8
horsepower	6
Unnamed: 9	406
Unnamed: 10	406
Unnamed: 11	406
Unnamed: 12	405

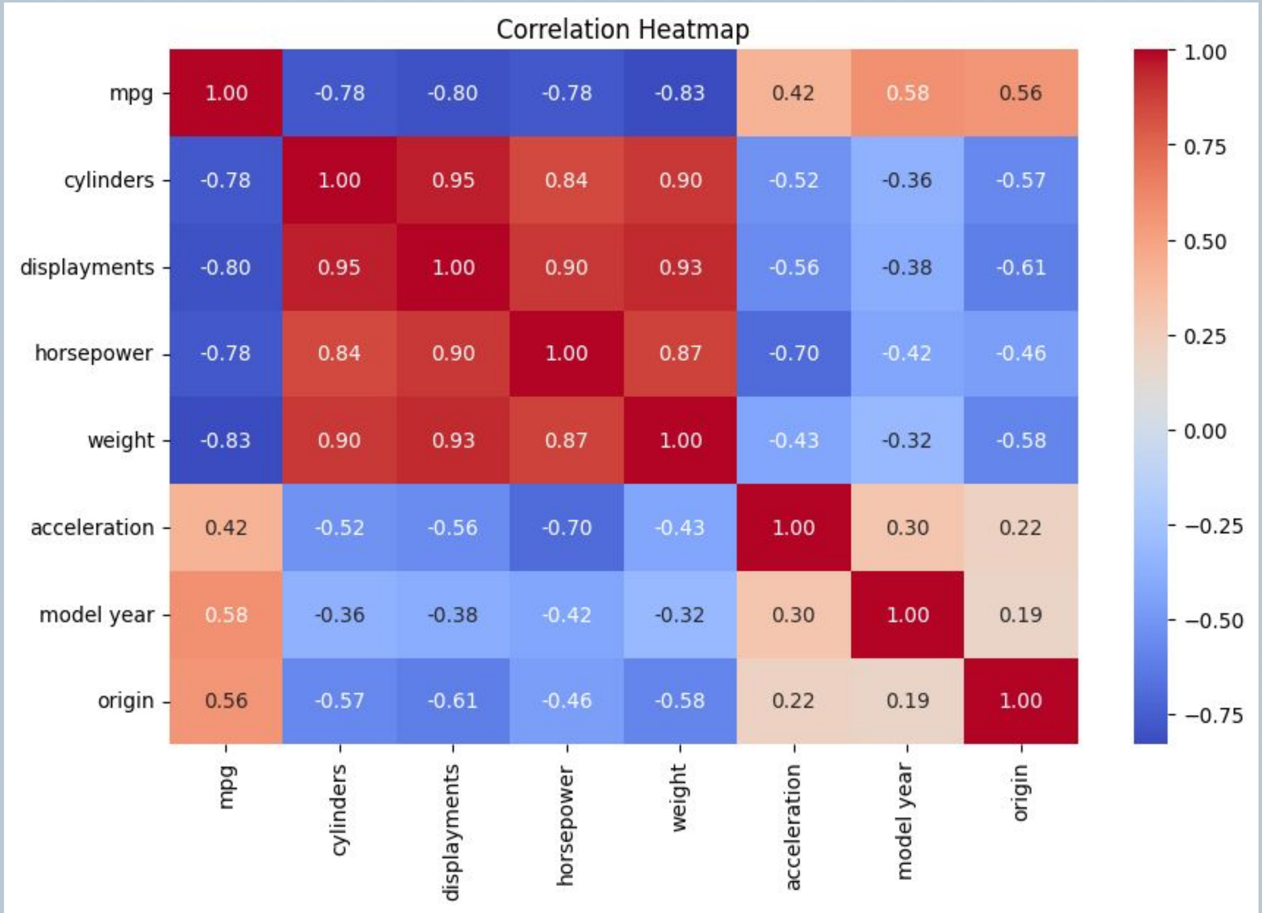
Histogram Results of Features



Insights:

- Most vehicles in the dataset have **moderate fuel efficiency, weight, and acceleration.**
- Cars with 4 cylinders **dominate**, and horsepower follows a bimodal trend.
- The model-years are evenly distributed, suggesting a **well-balanced dataset.**
- The origin of cars is clustered, with **some regions being more common** in the dataset.

Correlation Heatmap of Features



MPG: Negatively correlated with cylinders, displacement, horsepower, and weight

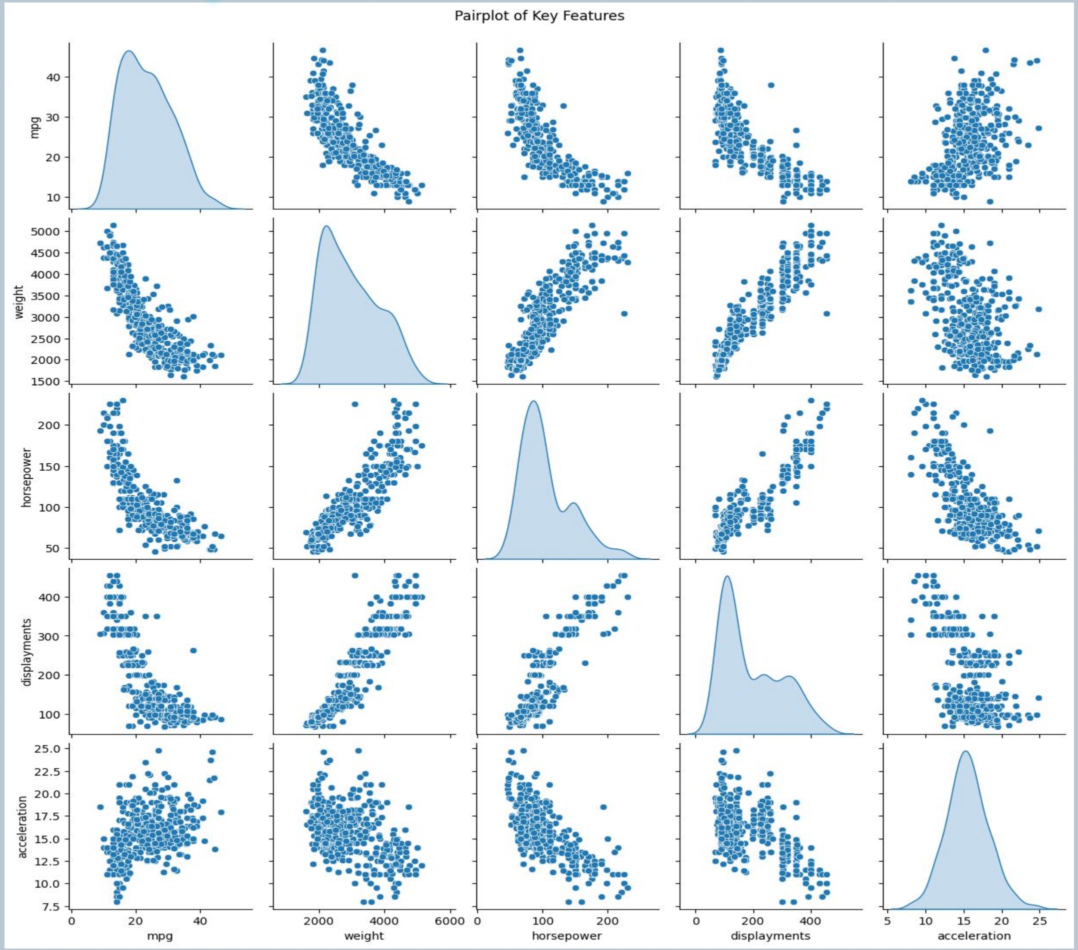
Cylinders, displacement, horsepower, weight: Strong intercorrelation, larger engines = heavier cars

Acceleration: Positively correlated with MPG

Model year: Positive correlation with MPG

Origin: Positive correlation with MPG

Pairplot Overview



MPG: Negative correlations with weight, horsepower, and displacement; positive correlation with acceleration

Engine-related features: Weight, horsepower, and displacement are strongly positively correlated

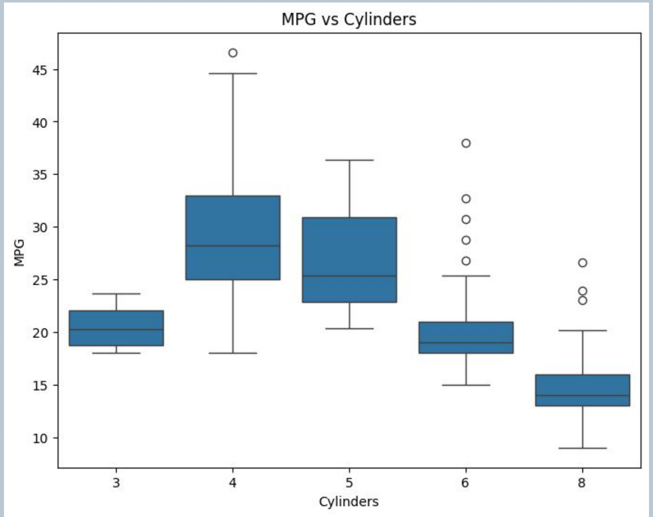
Insights:

Lighter cars with smaller engines are more fuel-efficient (higher MPG).

Heavier cars tend to have larger engines with more horsepower but consume more fuel.

Acceleration is influenced by both weight and engine power.

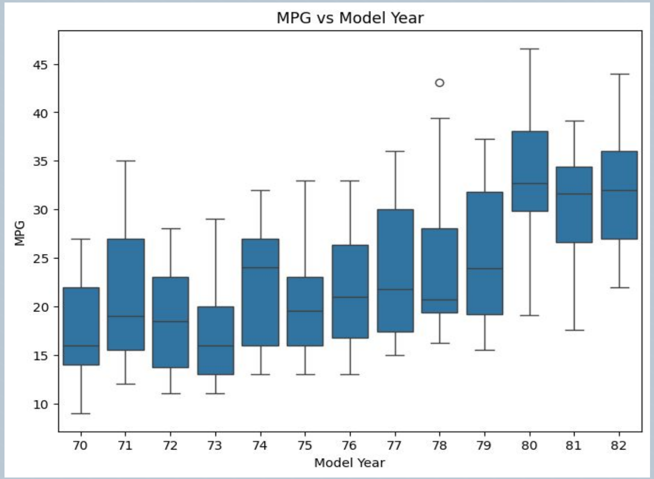
Boxplot Overview

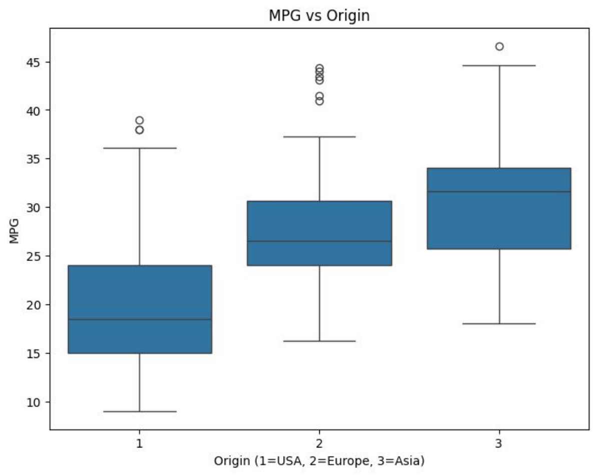


Fewer cylinders = higher mpg; more cylinders = lower mpg; outliers, especially in 4-cylinder cars with high mpg

Asia (3) has highest mpg, followed by Europe (2); USA (1) has lowest mpg; outliers in Europe and Asia with high mpg

MPG improves over the years, with early outliers showing higher values





Preprocessing - Standardizing car names

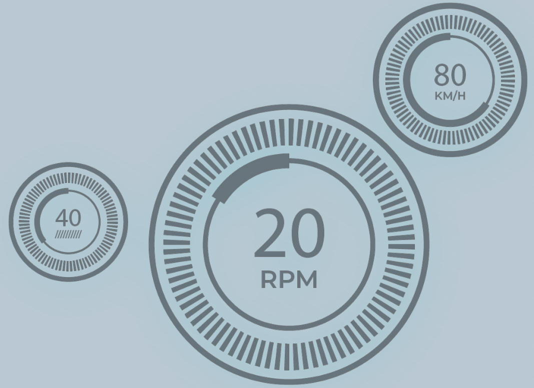
- Removed extra spaces and corrected misspellings.
- Extract brand function:Checks if the first two words form a known brand. If not, takes the first word as the brand.
- Known car brands: List of common car brands in lowercase.
- Extracted and standardized brand names.
- Brand corrections dictionary: Fixes typos and alternative brand names.

```
df['car name'] = df['car name'].str.replace(r'\s+', ' ', regex=True).str.strip()

known_brands = ["mercedes benz", "chevrolet", "ford", "buick", "plymouth",
    "volkswagen", "peugeot", "audi", "saab", "bmw", "toyota", "mazda", "pontiac",
    "mercury", "nissan", "honda", "subaru", "chrysler", "dodge", "fiat", "opel", "renault"]

def extract_brand(name):
    words = name.split()
    two_word_brand = " ".join(words[:2]).lower()
    if two_word_brand in known_brands:
        return two_word_brand
    return words[0].lower()

df['car_name_merge'] = df['car name'].apply(extract_brand)
```

```
# Define brand corrections
brand_corrections = {
    "mercedes": "mercedes-benz",
    "vw": "volkswagen",
    "chevy": "chevrolet",
    "toyouta": "toyota",
    "chevroelt": "chevrolet",
    "maxda": "mazda",
    "vokswagen": "volkswagen",
    "capri": "mercury",
    "datsun": "nissan"
}

df["car_name_merge"] = df["car_name_merge"].replace(brand_corrections)
```

03

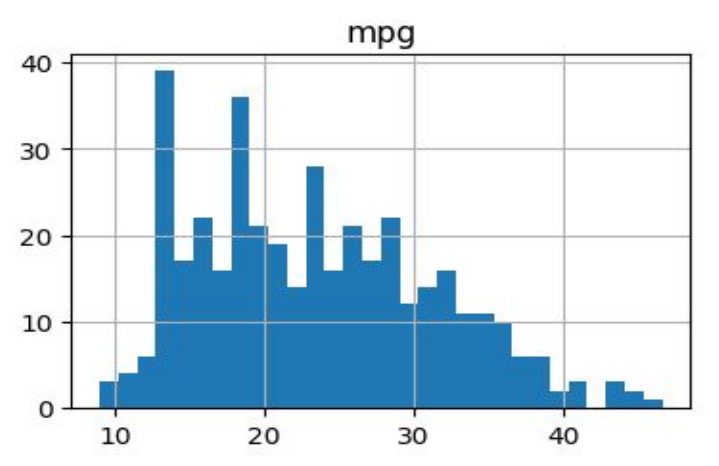
Training & Evaluation

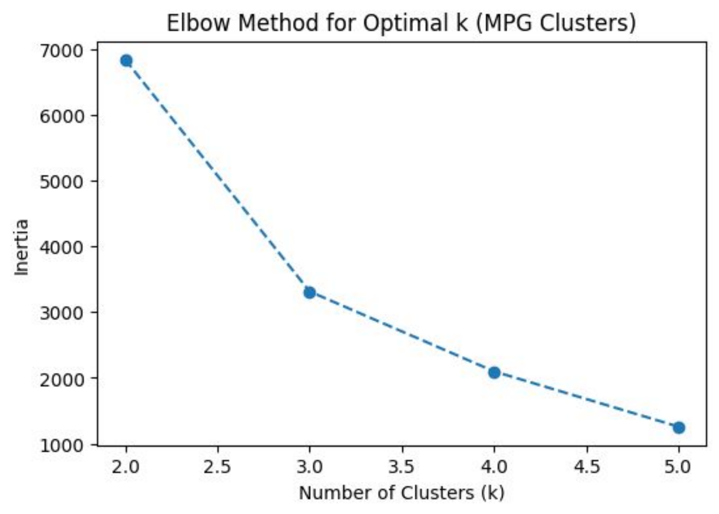


Fuel Efficiency Categories

- Manual calculation of MPG clusters based on observations of the mpg's histogram => k = 3

- KMeans Clustering. Based on the elbow diagram => k = 3





One hot encoding & Scaling

- Application of One Hot Encoding to car_name_merge, cylinders, origin columns

- Use of 2 Scalers:
    - StandardScaler()
    - MinMaxScaler()

- Creation of 2 datasets based on the used Scalers

Splitting of the datasets

- Standard dataset & min - max Dataset

- Split datasets using train_test_split()

```
# Splitting Standard Scaled Data
X_standard = df_standard_scaled.drop(columns=['mpg_cluster', 'mpg_kmeans'])
y_manual_standard = df_standard_scaled['mpg_cluster']
y_kmeans_standard = df_standard_scaled['mpg_kmeans']

X_train_manual_std, X_test_manual_std, y_train_manual_std, y_test_manual_std = train_test_split(
    X_standard, y_manual_standard, test_size=0.2, random_state=42, stratify=y_manual_standard
)
X_train_kmeans_std, X_test_kmeans_std, y_train_kmeans_std, y_test_kmeans_std = train_test_split(
    X_standard, y_kmeans_standard, test_size=0.2, random_state=42, stratify=y_kmeans_standard
)
```

Model evaluation and plotting

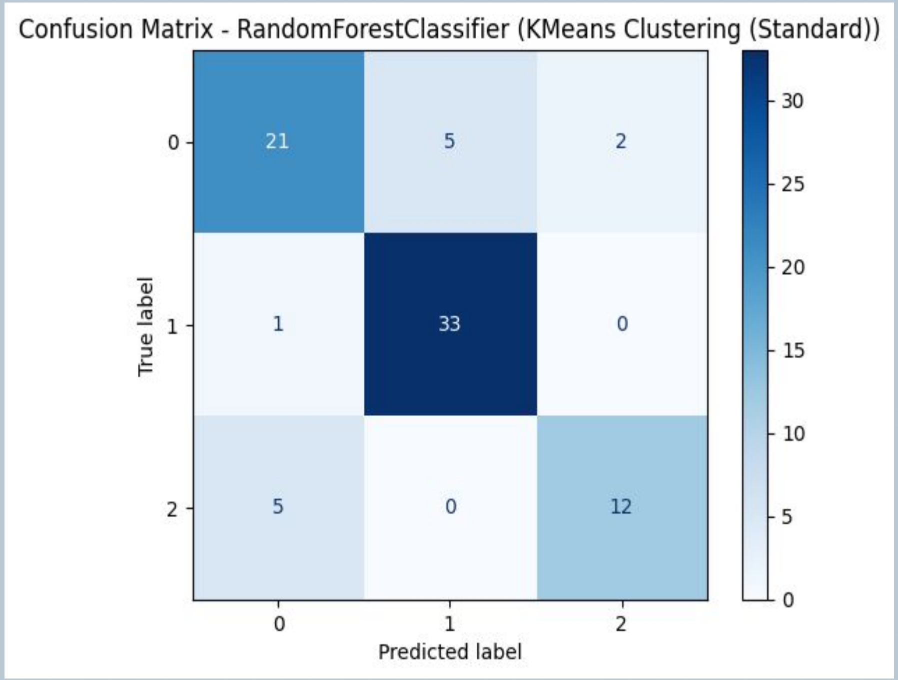
- Creation of a function for the evaluation and plotting of all the models
- Consideration of four variations of the dataset, based on different clustering methods (Manual vs. KMeans) and scaling techniques (Standard vs. MinMax)

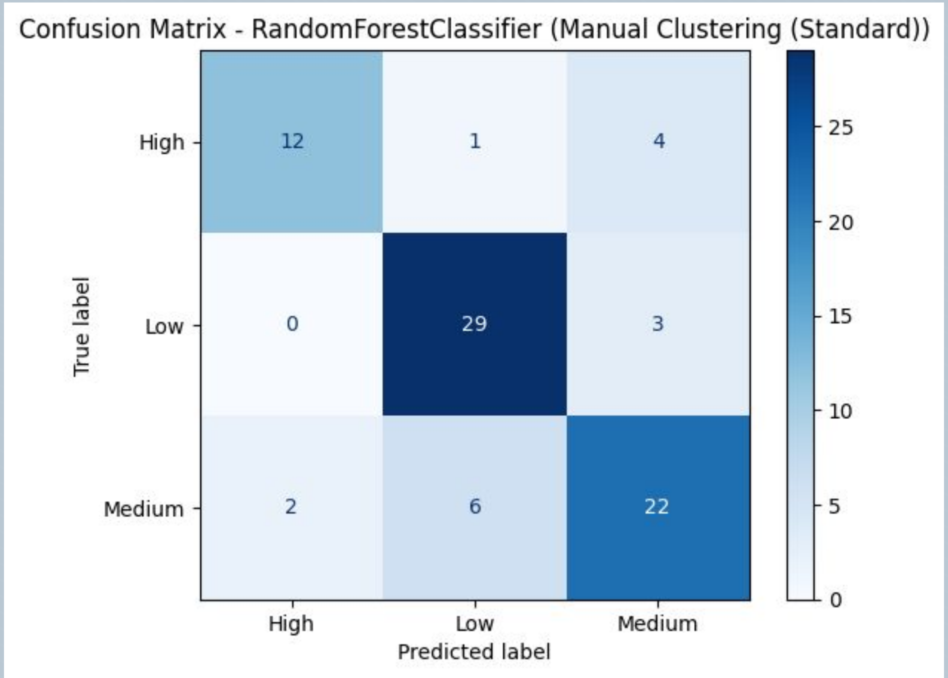
```python
def evaluate_and_plot_model(model, X_train, X_test, y_train, y_test, dataset_type):

    print(f"\n Training {model.__class__.__name__} on {dataset_type}...")
    model.fit(X_train, y_train)
    y_pred_test = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test, average="weighted")
    recall = recall_score(y_test, y_pred_test, average="weighted")
    f1 = f1_score(y_test, y_pred_test, average="weighted")


    print(f"\n {model.__class__.__name__} Performance ({dataset_type}):")
    print(f" Accuracy:  {accuracy:.4f}")
    print(f" Precision: {precision:.4f}")
    print(f" Recall:    {recall:.4f}")
    print(f" F1 Score:  {f1:.4f}")


    cm = confusion_matrix(y_test, y_pred_test)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_test))


    plt.figure(figsize=(6, 6))
    disp.plot(cmap="Blues", values_format=".0f")
    plt.title(f"Confusion Matrix - {model.__class__.__name__} ({dataset_type})")
```

```python
cm_path = os.path.join(plot_dir, f"confusion_matrix_{model.__class__.__name__}_{dataset_type}.png")
plt.savefig(cm_path, dpi=300)
print(f" Confusion matrix saved: {cm_path}")

plt.show()


return {
    "Model": model.__class__.__name__,
    "Dataset": dataset_type,
    "Accuracy": accuracy,
    "Precision": precision,
    "Recall": recall,
    "F1 Score": f1
}
```

Confusion Matrix

Prediction levels with Manual vs KMeans Clustering using Standard Scaler





Models Tested & Performance Evaluation

Model	Dataset	Accuracy	Precision	Recall	F1 Score
RandomForestClassifier	Manual Clustering (Standard)	0,797468	0,798833	0,797468	0,795295
RandomForestClassifier	Manual Clustering (MinMax)	0,797468	0,798833	0,797468	0,795295
SVC	Manual Clustering (Standard)	0,810127	0,828922	0,810127	0,80614
SVC	Manual Clustering (MinMax)	0,658228	0,518014	0,658228	0,57971
MLPClassifier	Manual Clustering (Standard)	0,734177	0,73219	0,734177	0,722429
MLPClassifier	Manual Clustering (MinMax)	0,721519	0,716189	0,721519	0,713643
KNeighborsClassifier	Manual Clustering (Standard)	0,772152	0,768655	0,772152	0,765249
KNeighborsClassifier	Manual Clustering (MinMax)	0,759494	0,760163	0,759494	0,752996
LogisticRegression	Manual Clustering (Standard)	0,734177	0,728748	0,734177	0,726878
LogisticRegression	Manual Clustering (MinMax)	0,759494	0,760759	0,759494	0,75211

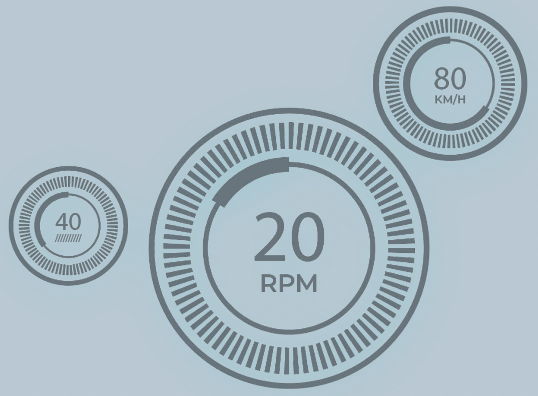
- KMeans generally improved performance for most models, especially for RandomForestClassifier and LogisticRegression.

Model	Dataset	Accuracy	Precision	Recall	F1 Score
RandomForestClassifier	KMeans Clustering (Standard)	0,835443	0,833867	0,835443	0,831769
RandomForestClassifier	KMeans Clustering (MinMax)	0,835443	0,833867	0,835443	0,831769
SVC	KMeans Clustering (Standard)	0,759494	0,759916	0,759494	0,752618
SVC	KMeans Clustering (MinMax)	0,658228	0,526063	0,658228	0,583821
MLPClassifier	KMeans Clustering (Standard)	0,746835	0,740617	0,746835	0,741318
MLPClassifier	KMeans Clustering (MinMax)	0,721519	0,72604	0,721519	0,722
KNeighborsClassifier	KMeans Clustering (Standard)	0,797468	0,795921	0,797468	0,79575
KNeighborsClassifier	KMeans Clustering (MinMax)	0,759494	0,754217	0,759494	0,751967
LogisticRegression	KMeans Clustering (Standard)	0,797468	0,795921	0,797468	0,79575
LogisticRegression	KMeans Clustering (MinMax)	0,797468	0,796154	0,797468	0,791072

- Standard scaling performed better overall, especially for SVC, KNeighborsClassifier, and LogisticRegression.

04

Prediction



Prediction

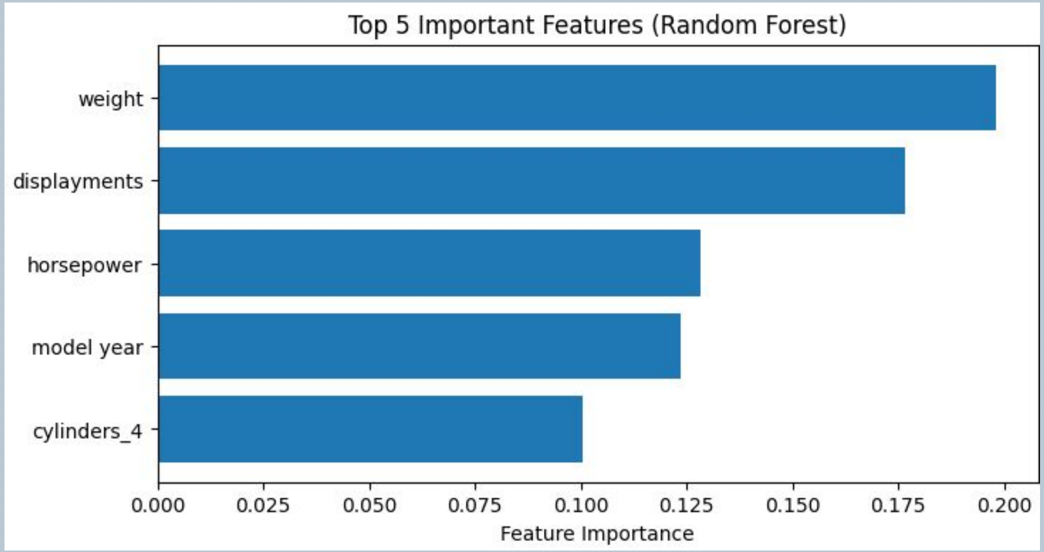
	Model	CV Accuracy	CV Precision	CV Recall	CV F1 Score
0	RandomForest	0.852227	0.854891	0.852227	0.851319
1	SVM	0.826677	0.835678	0.826677	0.822590
2	MLP	0.836303	0.839450	0.836303	0.835661
3	KNN	0.823502	0.824133	0.823502	0.822214
4	LogReg	0.845878	0.851615	0.845878	0.846371

- We used cross-validation to ensure reliable model performance by testing across multiple data splits. This reduces overfitting and improves generalization to unseen data.

- 5-fold cross-validation shows **Random Forest** and **Logistic Regression** consistently deliver the **strongest** performance across accuracy, precision, recall, and F1 score, indicating **robust generalisation.**

- All models demonstrate **stable performance**, evidenced by consistently high metrics, suggesting **reliable predictions on unseen data.**

Feature selection

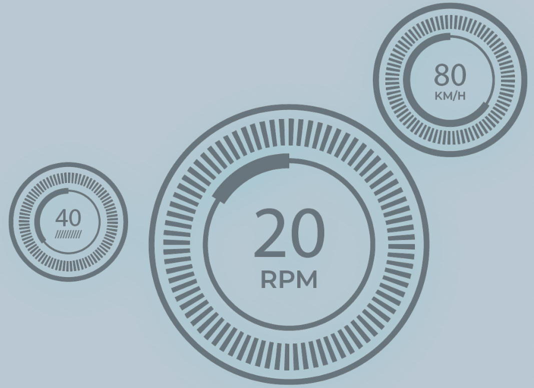


Random Forest Accuracy with Top 5 Features using CV: 0.8331

- **Weight** and **Displacement** are the most influential factors, indicating a strong correlation with the target variable.
- **Engine-related features** (horsepower and cylinders_4) along with **model year** also play significant roles.
- A combination of mechanical and temporal factors drive the model's outcome.
- The selected top 5 features align with the heatmap’s correlation trends

05

Conclusions



Conclusions

- **Successful Development of an MPG Prediction Model**
- **Effective Data Preprocessing and Feature Selection:** standardising car names and identifying key factors influencing MPG.
- **Robust Model Performance:** 5-fold cross-validation demonstrated that Random Forest and Logistic Regression models consistently delivered strong and stable performance.
- **Feature Importance Insights:** The most influential factors for MPG prediction were *weight*, *displacement*, and *engine-related* features.

Thank you for your attention!