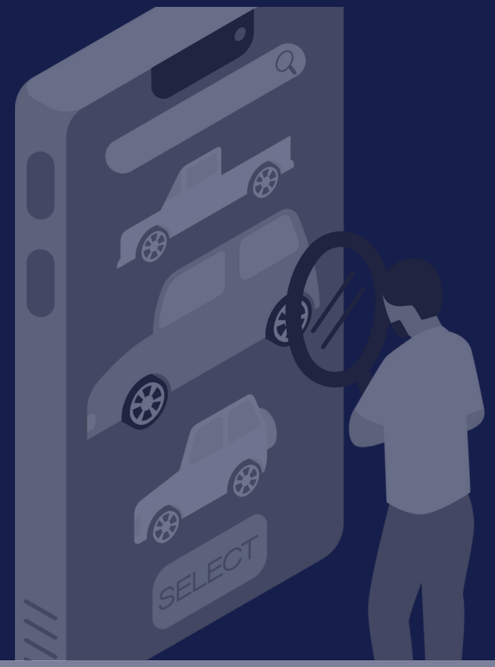
PREDICTING THE CITY-CYCLE FUEL CONSUMPTION



DATA SCIENCE & BUSINESS INTELLIGENCE BOOTCAMP

FEBRUARY 2025

Code.Hub

WE LEAD

MEET THE PROJECT TEAM

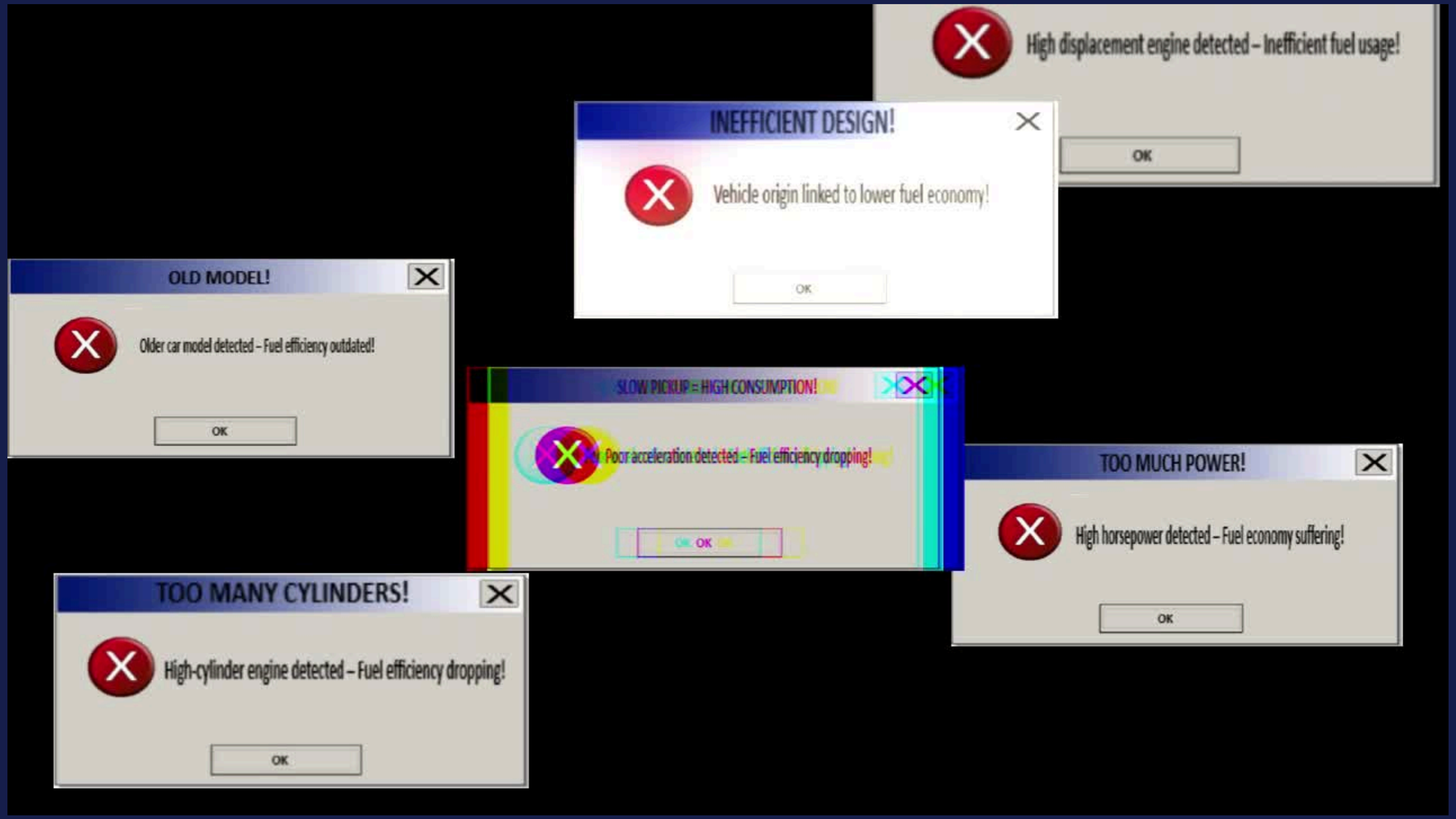
Dimitra Sykeridi
Data Analyst | PowerBI Specialist at PwC
BSc in Statistics and Insurance Science

Irida Alexandropoulou
MSc in Design of Digital Cultural Products
BSc in Cultural Technology and Communications UotA

Maria Kosmarika
MSc Law and Informatics,UOM x DUTH

Magda Sakanti
Senior Actuarial Consultant
MSc in Management Science and Technology, AUEB
BSc in Business Administration, AUEB

Angeliki Lagogianni
Project Manager|Agricultural Consultant
Integrated MSc in Food Science and Human Nutrition, AUA


High displacement engine detected – Inefficient fuel usage!
OK
INEFFICIENT DESIGN!
Vehicle origin linked to lower fuel economy!
OK
OLD MODEL!
Older car model detected – Fuel efficiency outdated!
OK
SLOW PICKUP = HIGH CONSUMPTION!
Poor acceleration detected – Fuel efficiency dropping!
OK
TOO MUCH POWER!
High horsepower detected – Fuel economy suffering!
OK
TOO MANY CYLINDERS!
High-cylinder engine detected – Fuel efficiency dropping!
OK

BUSINESS PROBLEM & SOLUTION

PROBLEM

Car rental companies struggle to balance cost-efficiency and customer satisfaction.
Fuel costs are a major operational expense.
No easy way to predict MPG for new or used fleet vehicles.

SOLUTION

A predictive analytics service that estimates MPG based on key vehicle attributes.

WETRUST

PROJECT GOALS

OBJECTIVE

Develop a Proof of Concept (POC) for a machine learning-based MPG prediction service.

PURPOSE

Assist car rental companies in making data-driven purchasing and pricing decisions.

BUSINESS IMPACT

- Optimize vehicle selection for better fuel efficiency.
- Reduce operational costs and environmental impact.
- Enhance customer satisfaction with fuel-efficient options.

WETRUST

DATASET OVERVIEW

Dataset Name: Auto MPG Dataset
No of Instances: 398
No of Attributes: 8 (excluding car name)
Attribute Types: Categorical & Numerical
Missing Values? Yes

Feature	Description	Unit
mpg	Fuel efficiency	MPG
Cylinders	Number of cylinders in the engine	Count
Displacement	Engine displacement	Cubic inches (in^3)
Horsepower	Engine power output	Horsepower (hp)
Weight	Vehicle weight	Pounds (lbs)
Acceleration	Time to reach 60 mph	Seconds
Model Year	Year of manufacturing	Year
Origin	Country of origin (1 = USA, 2 = Europe, 3 = Japan)	Categorical

PROJECT ROADMAP

A

🔍 **Understanding the Problem**

- Define objective (MPG classification)
- Explore dataset & features

B

📊 **EDA** - Python (Pandas, Numpy, Matplotlib, Seaborn)

- Visualize data trends
- Identify correlations & missing values

C

🛠️ **Data Preprocessing**

- Handle missing data & outliers
- Encode categorical features & scale data
- Categorize MPG into classes

D

🤖 **Model Development & Evaluation**

- Train multiple classification models
- Compare results

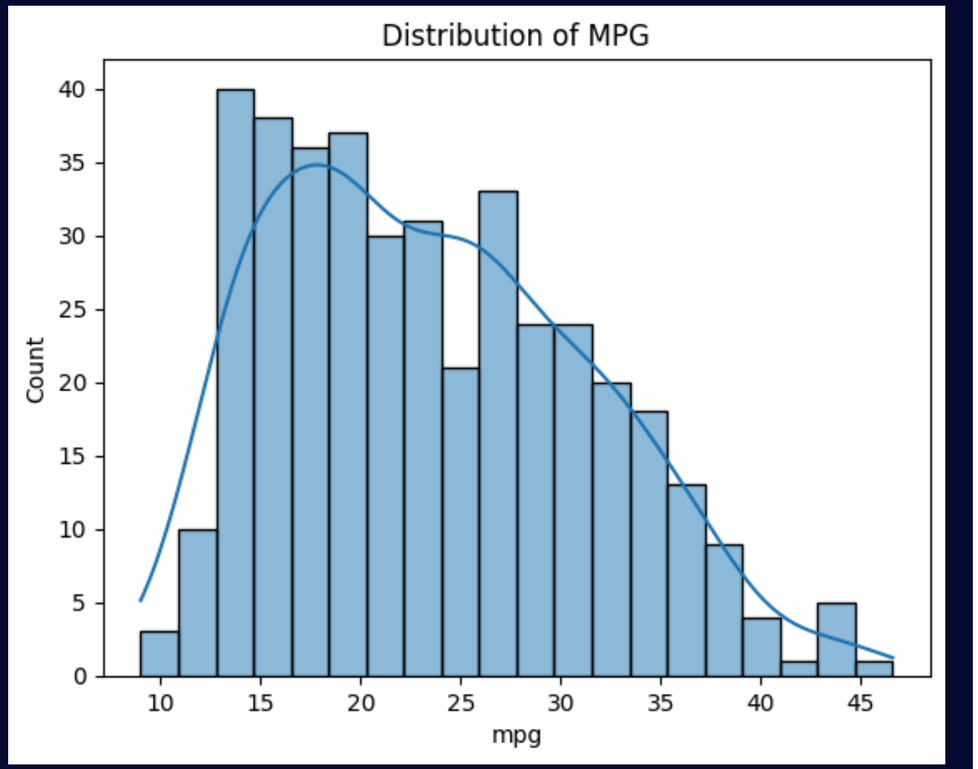
E

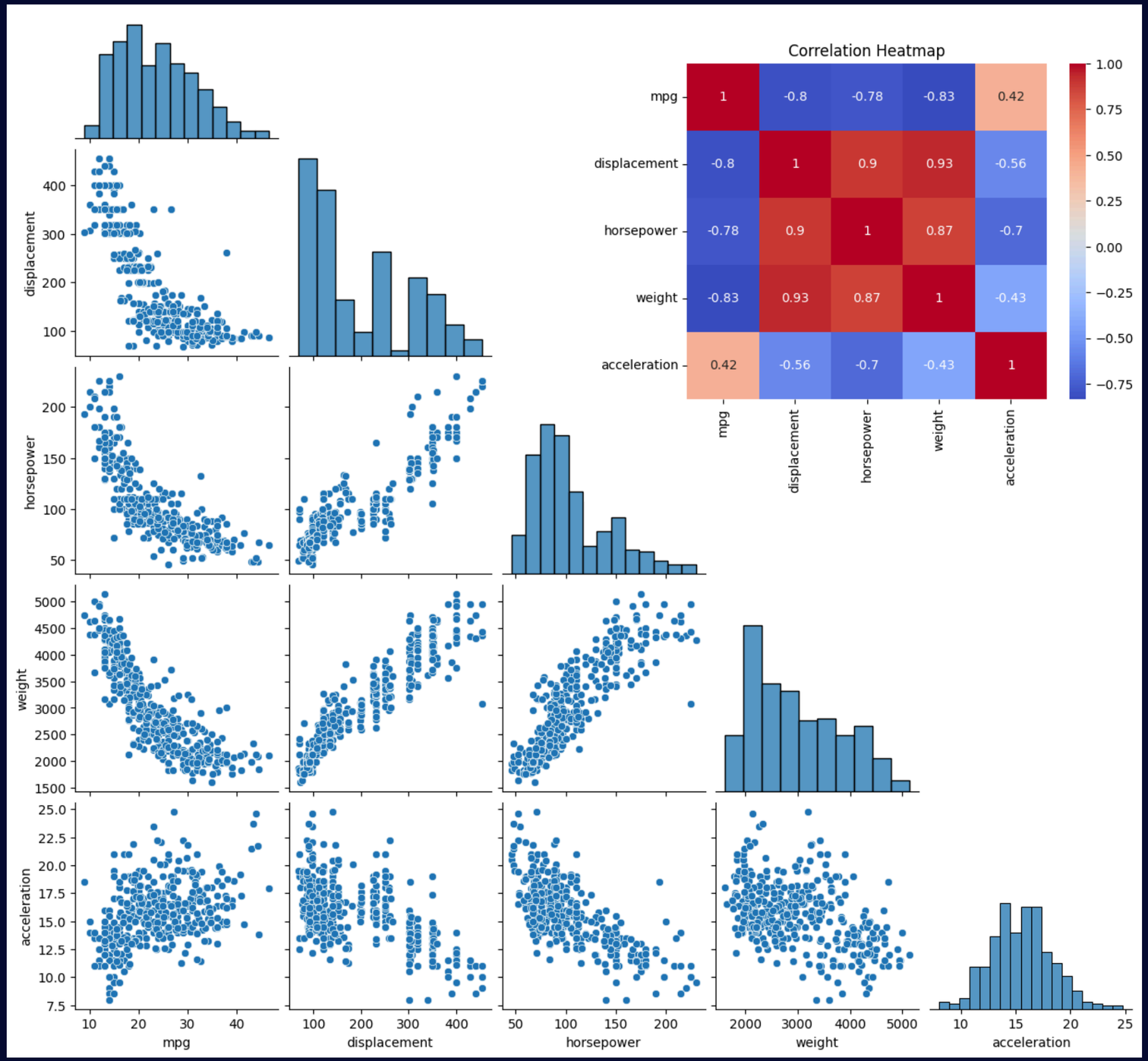
📈 **Model Fine-Tuning & Finalization**

- Select the best model
- Improve feature selection & performance

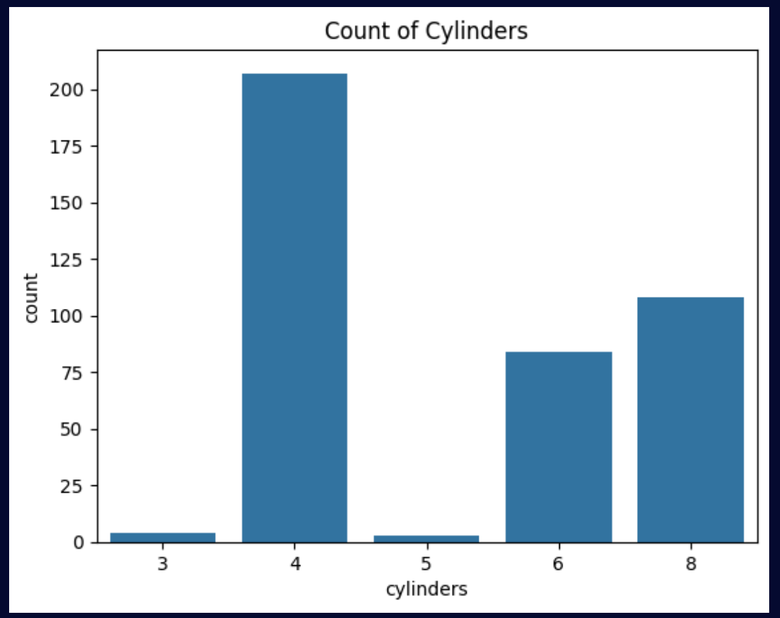
Deliver an MPG prediction service.

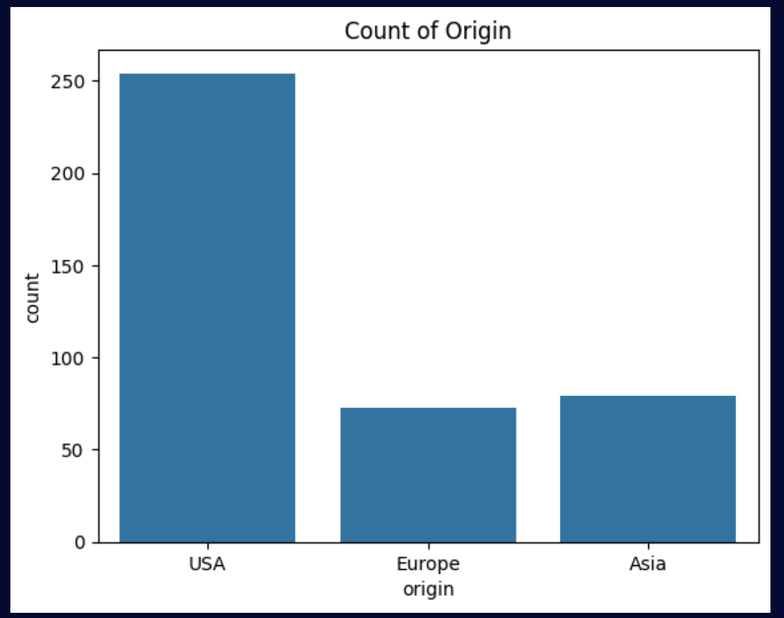
EDA

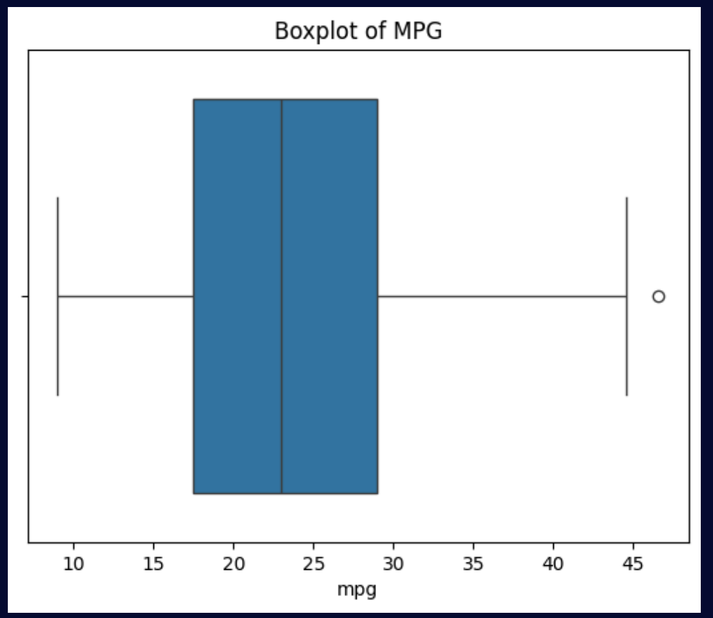

Distribution of MPG
Count
mpg
0
5
10
15
20
25
30
35
40
10
15
20
25
30
35
40
45

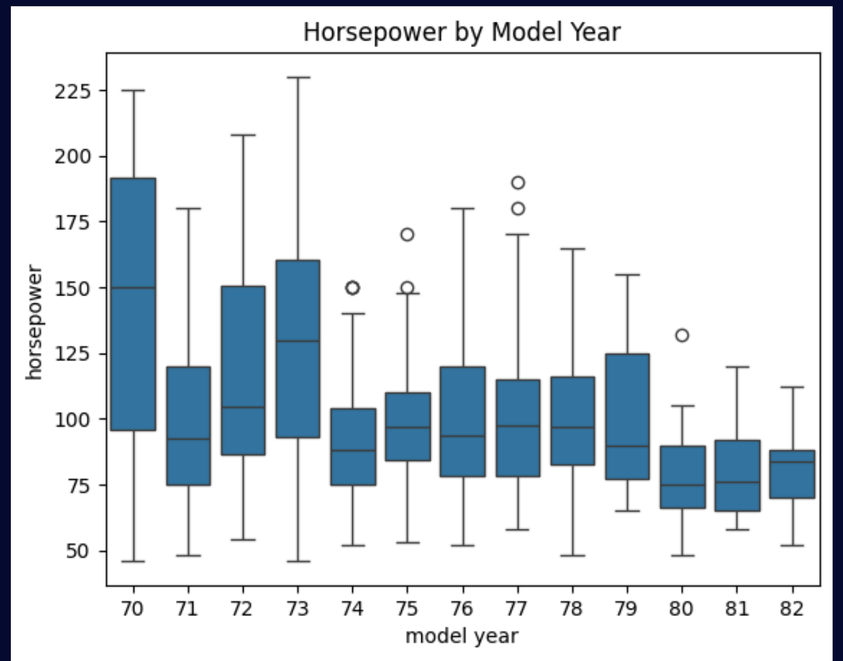

Correlation Heatmap
mpg
displacement
horsepower
weight
acceleration
1
-0.8
-0.78
-0.83
0.42
-0.8
1
0.9
0.93
-0.56
-0.78
0.9
1
0.87
-0.7
-0.83
0.93
0.87
1
-0.43
0.42
-0.56
-0.7
-0.43
1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
mpg
displacement
horsepower
weight
acceleration

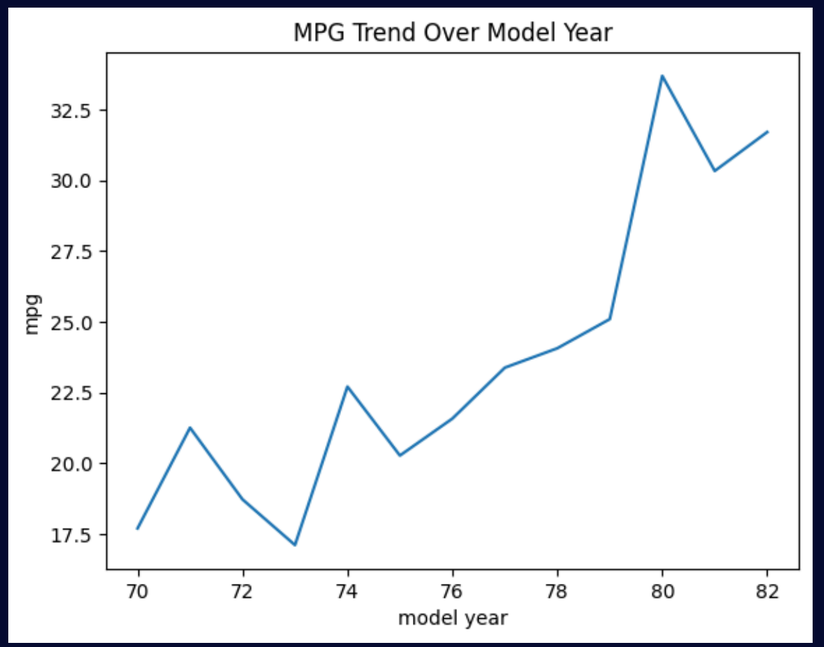
EDA


Count of Cylinders
count
cylinders
0
25
50
75
100
125
150
175
200
3
4
5
6
8


Count of Origin
count
origin
0
50
100
150
200
250
USA
Europe
Asia


Boxplot of MPG
mpg
10
15
20
25
30
35
40
45


Horsepower by Model Year
horsepower
model year
50
75
100
125
150
175
200
225
70
71
72
73
74
75
76
77
78
79
80
81
82


MPG Trend Over Model Year
mpg
model year
17.5
20.0
22.5
25.0
27.5
30.0
32.5
70
72
74
76
78
80
82

DATA PREPROCESSING

Drop unrelated columns

```
#Drop unrelated/blank columns
df.drop(columns=['Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11', 'Unnamed: 12'], axis=1, inplace=True)
```

Standardize data types

```
#change data type for displacements and horsepower
columns_for_change = ['displacements', 'horsepower']
df[columns_for_change] = df[columns_for_change].astype(int)
```

Resolve data inconsistencies

```
brand_corrections = {
    "vw": "volkswagen",
    "maxda": "mazda",
    "chevroelt": "chevrolet",
    "chevy": "chevrolet",
    "toyouta": "toyota",
    "mercedes": "mercedes-benz","vokswagen": "volkswagen","capri":"ford","mercury":"ford","triumph":"bmw","plymouth":"chrysler"}
mpg["brand"] = mpg["brand"].replace(brand_corrections)
```

Outlier detection & handling

```
column_name = "acceleration"

# Calculate z-scores for the column
mpg['z_score'] = zscore(mpg['acceleration'])
threshold = 3
mpg['outlier_zscore'] = np.abs(mpg['z_score']) > threshold

print("Outliers detected using Z-Score Method:")
print(mpg[mpg['outlier_zscore']])
```

```
Outliers detected using Z-Score Method:
      mpg  cylinders  displacements  horsepower  weight  acceleration  \
306  27.2          4            141          71    3190          24.8
402  44.0          4             97          52    2130          24.6

     model year  origin     car name    brand   model  brand_frequency  \
306        1979  Europe  peugeot 504  peugeot     504                8
402        1982  Europe    vw pickup       vw  pickup                6

      z_score  outlier_zscore
306  3.314505            True
402  3.243074            True
```

Create 2 new columns for brand and car name

```
#Split the Car Name Column to make two separate columns for brand and car name
df[['brand', 'model']] = df['car name'].str.split(' ', 1, expand=True)
df['model'] = df['model'].str.replace(' ', ' ', regex=False)
```

Check for duplicates

```
#check for duplicated values
df.duplicated().sum()
```

```
0
```

```
#check for null values
pd.isnull(df).sum()
```

```
mpg             8
cylinders       0
displayments    0
horsepower      6
weight          0
acceleration    0
model year      0
origin          0
car name        0
e: int64
```

Regression to classification

	mpg	cylinders	displayments	horsepower	weight	acceleration	model year	origin	car name	efficiency
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	3
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	3
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	3
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst	3
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino	3

MODEL DEVELOPMENT

Two Training Approaches:

- Without class weights
- With class weights (handling class imbalance)

Without class weights

Logistic Regression

```
Classification Report:
              precision    recall  f1-score   support

           1       0.86      0.75      0.80         8
           2       0.95      0.90      0.92        39
           3       0.92      1.00      0.96        33

    accuracy                           0.93        80
   macro avg       0.91      0.88      0.89        80
weighted avg       0.92      0.93      0.92        80
```

Decision Trees

```
Classification Report:
              precision    recall  f1-score   support

           1       0.50      0.50      0.50         8
           2       0.85      0.74      0.79        39
           3       0.84      0.97      0.90        33

    accuracy                           0.81        80
   macro avg       0.73      0.74      0.73        80
weighted avg       0.81      0.81      0.81        80
```

k-Nearest Neighbors (KNN)

```
Classification Report:
              precision    recall  f1-score   support

           1       0.89      1.00      0.94         8
           2       0.91      0.79      0.85        39
           3       0.81      0.91      0.86        33

    accuracy                           0.86        80
   macro avg       0.87      0.90      0.88        80
weighted avg       0.87      0.86      0.86        80
```

MODEL DEVELOPMENT

Two Training Approaches:

- Without class weights
- With class weights (handling class imbalance)

With class weights

Logistic Regression

```
Classification Report:
              precision    recall  f1-score   support

           1       0.57      1.00      0.73         8
           2       1.00      0.77      0.87        39
           3       0.92      1.00      0.96        33

    accuracy                           0.89        80
   macro avg       0.83      0.92      0.85        80
weighted avg       0.92      0.89      0.89        80
```

Decision Trees

```
Classification Report:
              precision    recall  f1-score   support

           1       0.43      0.38      0.40         8
           2       0.82      0.72      0.77        39
           3       0.82      0.97      0.89        33

    accuracy                           0.79        80
   macro avg       0.69      0.69      0.69        80
weighted avg       0.78      0.79      0.78        80
```

k-Nearest Neighbors (KNN)

```
Classification Report:
              precision    recall  f1-score   support

           1       0.89      1.00      0.94         8
           2       0.91      0.79      0.85        39
           3       0.81      0.91      0.86        33

    accuracy                           0.86        80
   macro avg       0.87      0.90      0.88        80
weighted avg       0.87      0.86      0.86        80
```

MODEL DEVELOPMENT

XGBoost

Without class weights

```
y_pred_xgb = xgb_model.predict(X_test)

print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))
```

```
XGBoost Accuracy: 0.8875
              precision    recall  f1-score   support

           0       0.86      0.75      0.80         8
           1       0.94      0.82      0.88        39
           2       0.85      1.00      0.92        33

    accuracy                           0.89        80
   macro avg       0.88      0.86      0.86        80
weighted avg       0.89      0.89      0.89        80
```

MODEL EVALUATION METRICS

	Model	Accuracy	Precision	Recall	F1-Score
Without weights	KNN	0.86	0.87	0.90	0.88
	Logistic Regression	0.93	0.91	0.88	0.89
	Decision Tree	0.81	0.73	0.74	0.73
With weights	KNN	0.86	0.87	0.90	0.88
	Logistic Regression	0.89	0.83	0.92	0.85
	Decision Tree	0.79	0.69	0.69	0.69
Without weights	XGBoost	0.89	0.89	0.89	0.89

Best Model Selection:

XGBoost outperformed other models in initial testing.

Selected for further cross-validation and fine-tuning.

CROSS-VALIDATION & FINE-TUNING

Cross-Validation: 5-fold

```
Cross Validation Scores: [0.766 0.859 0.844 0.778 0.841]
Mean Accuracy: 0.8175595238095237
Standard Deviation: 0.038148227552130 26
Το μοντέλο μπορεί να βελτιωθεί, ως προς την αξιολόγηση της ακρίβειας.
Σταθερό μοντέλο, ως προς την αξιολόγηση της διακύμανσης.
```

Grid Search

```
Best Parameters: {'C': 0.1, 'class_weight': None, 'penalty': 'l2', 'solver': 'saga'}
Optimized Logistic Regression Accuracy: 0.85
              precision    recall  f1-score   support

           1       0.75      0.38      0.50         8
           2       0.83      0.87      0.85        39
           3       0.89      0.94      0.91        33

    accuracy                           0.85        80
   macro avg       0.82      0.73      0.75        80
weighted avg       0.84      0.85      0.84        80
```

Cross-Validation (5-Fold)

- Used 5-Fold Cross-Validation (cv=5) to evaluate model performance.
- Calculated mean accuracy & standard deviation to assess stability.
- Interpreted results to determine if the model needed improvement.

Hyperparameter Tuning (Grid Search CV)

- Defined a parameter gridfor tuning:

-C: Regularization strength (0.01, 0.1, 1, 10, 100).
-solver: Optimization solvers (liblinear, saga).
-penalty: Regularization type (l1, l2).
-class_weight: Handling of imbalanced data (None, 'balanced').

- Used GridSearchCV with 5-Fold CV to find the best hyperparameters.
- Trained the model with optimal hyperparameters.

Model Evaluation on Test Set

- Used the best model (best_lr_model) from Grid Search.
- Predicted on the test set (y_pred_best = best_lr_model.predict(X_logistic_test)).
- Evaluated model using:

-Accuracy (accuracy_score)
-Precision, Recall, and F1-score (classification_report)

Final Recommendations:

1. Deploy the Logistic Regression model for production use

2. Further work on XGBoost

CONCLUSION

- ✓ Successfully reframed the regression problem into classification.
- ✓ Preprocessed data effectively for machine learning.
- ✓ Tested multiple models and fine-tuned the best performer.
- ✓ Provided a robust and scalable solution for predicting MPG categories.

Key Insights:

- High-consumption cars tend to have higher weight, displacement, and horsepower.
- Low-consumption cars are generally lighter and have smaller engines.
- Model year impacts fuel efficiency—newer cars are often more efficient.

WETRUST

CHALLENGES & FUTURE STEPS

MODEL INTERPRETABILITY

- More complex models like XGBoost might be harder to explain to stakeholders.
- Exploring explainability methods (e.g., SHAP, LIME).

SCALABILITY

- The dataset used is small (398 instances), but real-world applications may deal with much larger datasets.
- Ensuring that models can scale effectively for larger datasets.

MODEL OPTIMIZATION FOR PRODUCTION

- Transitioning from research models to production-ready solutions, including monitoring model drift and retraining over time.

FEATURE ENGINEERING ENHANCEMENTS

1. Explore additional features (e.g., engine size, fuel type) to improve model accuracy.
2. Investigate interactions between features for better insights.
3. Incorporate real-time driving conditions for better classification.

FURTHER MODEL OPTIMIZATION

1. Test advanced algorithms like Random Forest, SVM, or Neural Networks.
2. Experiment with ensemble learning (e.g., stacking models for better generalization).

DEPLOYMENT & REAL-WORLD INTEGRATION

1. Deploy the model using Flask or FastAPI for real-time predictions.
2. Develop an interactive dashboard for car rental companies to visualize predictions.

DO YOU HAVE ANY QUESTIONS?

Magda Sakanti

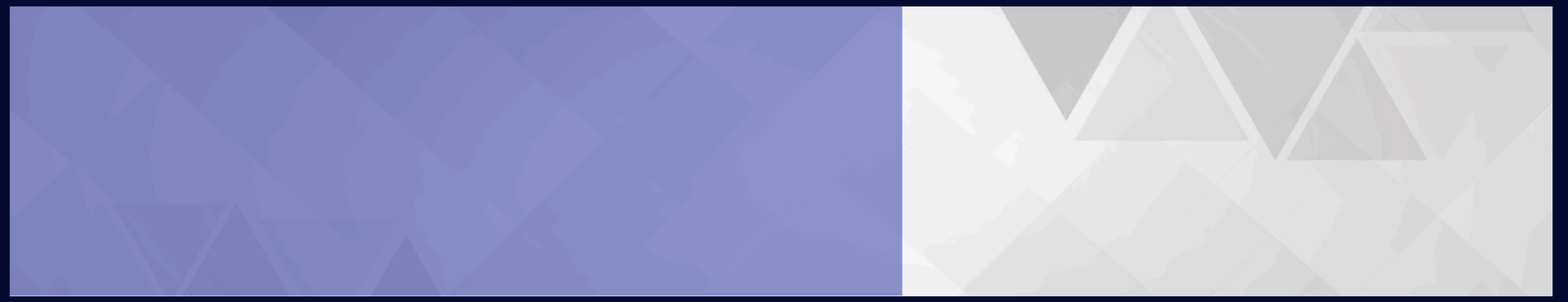
Maria Kosmarika

Irida Alexandropoulou

Angeliki Lagogianni

Dimitra Sykeridi

SCAN ME




WETRUST