**Car Price Prediction with Machine Learning**

**Objective**

The objective of this project is to predict the selling price of cars based on various features such as the car's brand, features, horsepower, mileage, and more. Predicting car prices accurately can help buyers and sellers make informed decisions, thereby facilitating better negotiations and transactions.

**Solution**

To achieve the objective, we employ machine learning techniques to build a predictive model. The solution involves data preprocessing, exploratory data analysis (EDA), feature engineering, model training and evaluation, hyperparameter tuning, feature importance analysis, and model saving. We utilize Python and several of its libraries, including Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, and Joblib.

**Procedure**

**Step 1: Load the Data**

We start by loading the dataset using Pandas and examining its structure, missing values, and summary statistics.

import pandas as pd

file\_path = '/mnt/data/car data.csv'

car\_data = pd.read\_csv(file\_path)

print(car\_data.head())

print(car\_data.info())

print(car\_data.describe())

**Step 2: Data Preprocessing**

Data preprocessing involves handling missing values, encoding categorical variables, scaling numerical features, and creating new features.

1. **Handle missing values**: We drop rows with missing values.
2. **Encode categorical variables**: Categorical variables such as 'Fuel\_Type', 'Selling\_type', and 'Transmission' are encoded using OneHotEncoder.
3. **Scale numerical features**: Numerical features such as 'Present\_Price', 'Driven\_kms', 'Owner', and 'Car\_Age' are scaled using StandardScaler.
4. **Create new features**: We create a new feature 'Car\_Age' by subtracting the year of manufacture from the current year.

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

car\_data = car\_data.dropna()

car\_data['Car\_Age'] = 2023 - car\_data['Year']

car\_data = car\_data.drop('Year', axis=1)

categorical\_features = ['Fuel\_Type', 'Selling\_type', 'Transmission']

numerical\_features = ['Present\_Price', 'Driven\_kms', 'Owner', 'Car\_Age']

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numerical\_features),

('cat', OneHotEncoder(), categorical\_features)

])

**Step 3: Exploratory Data Analysis (EDA)**

EDA helps us understand the data distribution and relationships between variables. We visualize the distribution of the target variable 'Selling\_Price' and the relationships between features and the target variable.

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10, 6))

sns.histplot(car\_data['Selling\_Price'], kde=True)

plt.title('Distribution of Selling Price')

plt.xlabel('Selling Price')

plt.ylabel('Frequency')

plt.show()

sns.pairplot(car\_data, x\_vars=numerical\_features, y\_vars='Selling\_Price', height=5, aspect=0.7)

plt.show()

**Step 4: Feature Engineering**

We update the list of numerical features to include the new 'Car\_Age' feature.

numerical\_features = ['Present\_Price', 'Driven\_kms', 'Owner', 'Car\_Age']

**Step 5: Model Training**

We split the data into training and testing sets and train two models: Linear Regression and Random Forest. The models are evaluated based on Mean Squared Error (MSE) and R-squared (R2) scores.

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import Pipeline

from sklearn.metrics import mean\_squared\_error, r2\_score

X = car\_data.drop('Selling\_Price', axis=1)

y = car\_data['Selling\_Price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

models = {

'Linear Regression': LinearRegression(),

'Random Forest': RandomForestRegressor(random\_state=42)

}

results = {}

for model\_name, model in models.items():

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

('model', model)])

pipeline.fit(X\_train, y\_train)

y\_pred = pipeline.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

results[model\_name] = {'MSE': mse, 'R2': r2}

for model\_name, metrics in results.items():

print(f"{model\_name} - MSE: {metrics['MSE']:.2f}, R2: {metrics['R2']:.2f}")

**Step 6: Hyperparameter Tuning for Random Forest**

We use GridSearchCV to find the best hyperparameters for the Random Forest model.

from sklearn.model\_selection import GridSearchCV

param\_grid = {

'model\_\_n\_estimators': [50, 100, 200],

'model\_\_max\_depth': [None, 10, 20, 30],

'model\_\_min\_samples\_split': [2, 5, 10],

'model\_\_min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(

estimator=Pipeline(steps=[('preprocessor', preprocessor),

('model', RandomForestRegressor(random\_state=42))]),

param\_grid=param\_grid,

cv=5,

n\_jobs=-1,

scoring='neg\_mean\_squared\_error'

)

grid\_search.fit(X\_train, y\_train)

print(f"Best parameters: {grid\_search.best\_params\_}")

print(f"Best score (neg MSE): {grid\_search.best\_score\_}")

best\_model = grid\_search.best\_estimator\_

y\_pred = best\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Random Forest (Best Model) - MSE: {mse:.2f}, R2: {r2:.2f}")

**Step 7: Feature Importance Analysis**

We analyze and visualize the feature importances from the best Random Forest model.

importances = best\_model.named\_steps['model'].feature\_importances\_

features = numerical\_features + list(grid\_search.best\_estimator\_.named\_steps['preprocessor'].named\_transformers\_['cat'].get\_feature\_names\_out(categorical\_features))

feature\_importance\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

feature\_importance\_df = feature\_importance\_df.sort\_values(by='Importance', ascending=False)

plt.figure(figsize=(12, 8))

sns.barplot(x='Importance', y='Feature', data=feature\_importance\_df)

plt.title('Feature Importances')

plt.show()

**Step 8: Save the Best Model**

We save the best model for future use.

import joblib

joblib.dump(best\_model, 'best\_random\_forest\_model.pkl')

**Output**

1 to 5 of 5 entriesFilter

| **index** | **Car\_Name** | **Year** | **Selling\_Price** | **Present\_Price** | **Driven\_kms** | **Fuel\_Type** | **Selling\_type** | **Transmission** | **Owner** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | ritz | 2014 | 3.35 | 5.59 | 27000 | Petrol | Dealer | Manual | 0 |
| **1** | sx4 | 2013 | 4.75 | 9.54 | 43000 | Diesel | Dealer | Manual | 0 |
| **2** | ciaz | 2017 | 7.25 | 9.85 | 6900 | Petrol | Dealer | Manual | 0 |
| **3** | wagon r | 2011 | 2.85 | 4.15 | 5200 | Petrol | Dealer | Manual | 0 |
| **4** | swift | 2014 | 4.6 | 6.87 | 42450 | Diesel | Dealer | Manual | 0 |

Show 102550100 per page

Like what you see? Visit the [data table notebook](https://colab.research.google.com/notebooks/data_table.ipynb) to learn more about interactive tables.

**Distributions**

A graph of blue bars

Description automatically generated with medium confidence

A graph of a number of blue bars

Description automatically generated with medium confidence

A graph of blue vertical bars

Description automatically generated

A graph of blue vertical bars

Description automatically generated

**Categorical distributions**

A colorful bars with numbers

Description automatically generated

A graph with a bar and a number

Description automatically generated with medium confidence

**2-d distributions**

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

A graph with blue dots

Description automatically generated

**Time series**

A graph with numbers and lines

Description automatically generated

A line graph with a green line

Description automatically generated

A graph with numbers and lines

Description automatically generated

A line graph with orange and green lines

Description automatically generated

**Values**

A blue line graph with white text

Description automatically generated

A graph with a line

Description automatically generated

A line graph with text

Description automatically generated

A graph with a line

Description automatically generated

**2-d categorical distributions**

A chart of different colored squares

Description automatically generated with medium confidence

**Faceted distributions**

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

A line graph with numbers

Description automatically generated with medium confidence

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

A green and orange rectangular object

Description automatically generated

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

A white background with black lines

Description automatically generated

<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

A green and orange object

Description automatically generated with medium confidence

A graph of a distribution of selling price

Description automatically generatedA graph of a person and person

Description automatically generated with medium confidence

Linear Regression - MSE: 3.48, R2: 0.85

Random Forest - MSE: 0.90, R2: 0.96

Best parameters: {'model\_\_max\_depth': None, 'model\_\_min\_samples\_leaf': 1, 'model\_\_min\_samples\_split': 2, 'model\_\_n\_estimators': 100}

Best score (neg MSE): -2.9611308159583336

Random Forest (Best Model) - MSE: 0.90, R2: 0.96

A graph with blue squares

Description automatically generated with medium confidence['best\_random\_forest\_model.pkl']

**Conclusion**

In this project, we successfully built a machine learning model to predict car prices based on various features. We explored the data, preprocessed it, engineered features, trained and evaluated models, performed hyperparameter tuning, analyzed feature importances, and saved the best model.

The Random Forest model with tuned hyperparameters achieved the best performance with an MSE of X.XX and an R2 score of X.XX on the test set. The most important features influencing car prices were found to be [list top features].

This project demonstrates the end-to-end process of building a predictive model, including data preprocessing, model training, hyperparameter tuning, and evaluation. The resulting model can be used to make accurate car price predictions, assisting buyers and sellers in making informed decisions.

This report includes the objectives, solution approach, detailed procedure, theoretical background, algorithms used, code execution, and conclusion. You can expand on specific sections with more details if needed.