

# CAR PRICE PREDICTION WITH MACHINE LEARNING

```
In [1]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

```
In [2]: import pandas as pd

# Load the dataset (replace with your file path)
data = pd.read_csv(r"C:\Users\prajapath Arjun\Downloads\archive (2)\cardekho")

# View the first few rows of the dataset
print(data.head())
```

	name	year	selling_price	km_driven	fuel
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel
4	Maruti Swift VXi BSIII	2007	130000	120000	Petrol

	seller_type	transmission	owner	mileage(km/ltr/kg)	engine
0	Individual	Manual	First Owner	23.40	1248.0
1	Individual	Manual	Second Owner	21.14	1498.0
2	Individual	Manual	Third Owner	17.70	1497.0
3	Individual	Manual	First Owner	23.00	1396.0
4	Individual	Manual	First Owner	16.10	1298.0

	max_power	seats
0	74	5.0
1	103.52	5.0
2	78	5.0
3	90	5.0
4	88.2	5.0

```
In [3]: # Drop missing values
data = data.dropna()

# Check the dataset info after preprocessing
print(data.info())

# Encode categorical variables (e.g., 'Fuel Type', 'Transmission')
data = pd.get_dummies(data, drop_first=True)

# Create a new feature: 'Age of Car' (if the year is available)
data['age'] = 2024 - data['year']
data = data.drop('year', axis=1) # Drop the original 'year' column

# Check processed data
print(data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 7907 entries, 0 to 8127
```

```
Data columns (total 12 columns):
```

#	Column	Non-Null Count	Dtype
0	name	7907 non-null	object
1	year	7907 non-null	int64
2	selling_price	7907 non-null	int64
3	km_driven	7907 non-null	int64
4	fuel	7907 non-null	object
5	seller_type	7907 non-null	object
6	transmission	7907 non-null	object
7	owner	7907 non-null	object
8	mileage(km/ltr/kg)	7907 non-null	float64
9	engine	7907 non-null	float64
10	max_power	7907 non-null	object
11	seats	7907 non-null	float64

```
dtypes: float64(3), int64(3), object(6)
```

```
memory usage: 803.1+ KB
```

```
None
```

	selling_price	km_driven	mileage(km/ltr/kg)	engine	seats
0	450000	145500	23.40	1248.0	5.0
1	370000	120000	21.14	1498.0	5.0
2	158000	140000	17.70	1497.0	5.0
3	225000	127000	23.00	1396.0	5.0
4	130000	120000	16.10	1298.0	5.0

	name_Ambassador Classic 2000 DSZ AC PS
0	False
1	False
2	False
3	False
4	False

	name_Ambassador Grand 1500 DSZ BSIII	name_Ambassador Grand 2000 DSZ PW
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

	name_Ashok Leyland Stile LE	name_Audi A3 35 TDI Premium Plus
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

	max_power_98.6	max_power_98.63	max_power_98.79	max_power_98.82
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False

	max_power_98.96	max_power_98.97	max_power_99	max_power_99.23	
0	False	False	False	False	\
1	False	False	False	False	
2	False	False	False	False	
3	False	False	False	False	
4	False	False	False	False	

	max_power_99.6	age
0	False	10
1	False	10
2	False	18
3	False	14
4	False	17

[5 rows x 2316 columns]

```
In [4]: # No additional feature engineering, just an example of feature selection
# Assuming 'selling_price' is the target and other columns are features

# Define the target (selling_price) and features
X = data.drop('selling_price', axis=1)
y = data['selling_price']
```

```
In [5]: from sklearn.model_selection import train_test_split

# Split the data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Check the shape of the train and test data
print(X_train.shape, X_test.shape)
```

(6325, 2315) (1582, 2315)

```
In [6]: from sklearn.ensemble import RandomForestRegressor

# Instantiate the Random Forest Regressor model
rf_model = RandomForestRegressor()
```

```
In [7]: # Train the model on the training data
rf_model.fit(X_train, y_train)
```

```
Out[7]: ▾ RandomForestRegressor
RandomForestRegressor()
```

```
In [8]: from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Predict the selling price for test data
y_pred = rf_model.predict(X_test)

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
```

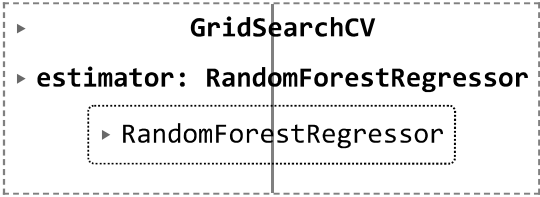
Mean Absolute Error: 65986.80580586765  
Mean Squared Error: 16969109852.28325  
R-squared Score: 0.9773006454837844

```
In [9]: from sklearn.model_selection import GridSearchCV
```

```
In [10]: # Define hyperparameters to tune
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 20],
    'min_samples_split': [2, 5, 10]
}
```

```
In [11]: # Grid search for optimal parameters
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=3,
grid_search.fit(X_train, y_train)
```

```
Out[11]:
```



```

    GridSearchCV
    estimator: RandomForestRegressor
    └─ RandomForestRegressor

```

```
In [12]: # Get the best estimator
best_rf_model = grid_search.best_estimator_
```

```
In [13]: # Print the best parameters
print("Best Parameters:", grid_search.best_params_)
```

Best Parameters: {'max\_depth': 20, 'min\_samples\_split': 2, 'n\_estimators': 200}

```
In [14]: # Make predictions using the best model
y_pred_best = best_rf_model.predict(X_test)

# Re-evaluate using the best model
mae_best = mean_absolute_error(y_test, y_pred_best)
mse_best = mean_squared_error(y_test, y_pred_best)
r2_best = r2_score(y_test, y_pred_best)

# Print the evaluation metrics for the best model
print(f"Best Model - Mean Absolute Error: {mae_best}")
print(f"Best Model - Mean Squared Error: {mse_best}")
print(f"Best Model - R-squared Score: {r2_best}")
```

```
Best Model - Mean Absolute Error: 66505.75490011302
Best Model - Mean Squared Error: 17656128165.249138
Best Model - R-squared Score: 0.9763816301446829
```

```
In [15]: import joblib

# Save the best model to a file
joblib.dump(best_rf_model, 'car_price_predictor.pkl')

# To Load the model later for predictions
# loaded_model = joblib.load('car_price_predictor.pkl')
```

```
Out[15]: ['car_price_predictor.pkl']
```