## Import Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import string as str

df = pd.read_csv('used_car_price.csv')
```

```
df.head()
```

| | make | model | price_usd | year | kilometer | fuel_type | transmission | color | owner |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Honda | Amaze 1.2 VX i-VTEC | 6060 | 2017 | 87150 | Petrol | Manual | Grey | First |
| **1** | Maruti Suzuki | Swift DZire VDI | 5400 | 2014 | 75000 | Diesel | Manual | White | Second |
| **2** | Hyundai | i10 Magna 1.2 Kappa2 | 2640 | 2011 | 67000 | Petrol | Manual | Maroon | First |
| **3** | Toyota | Glanza G | 9588 | 2019 | 37500 | Petrol | Manual | Red | First |
| **4** | Toyota | Innova 2.4 VX 7 STR [2016-2020] | 23400 | 2018 | 69000 | Diesel | Manual | Grey | First |

Next steps: View recommended plots

## Import Libraries

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
```

## Data Preparation

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2059 entries, 0 to 2058
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   make                 2059 non-null   object
 1   model                2059 non-null   object
 2   price_usd            2059 non-null   int64
 3   year                 2059 non-null   int64
 4   kilometer            2059 non-null   int64
 5   fuel_type            2059 non-null   object
 6   transmission         2059 non-null   object
 7   color                2059 non-null   object
 8   owner                2059 non-null   object
 9   seller_type          2059 non-null   object
 10  engine               1979 non-null   object
 11  max_power            1979 non-null   object
 12  max_torque           1979 non-null   object
 13  drivetrain           1923 non-null   object
 14  length_mm            1995 non-null   float64
 15  width_mm             1995 non-null   float64
 16  height_mm            1995 non-null   float64
 17  seating_capacity     1995 non-null   float64
 18  fuel_tank_capacity_l 1946 non-null   float64
```

```
dtypes: float64(5), int64(3), object(11)
memory usage: 305.8+ KB
```

Find Missing Values

```
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])
```

```
engine                  80
max_power               80
max_torque              80
drivetrain             136
length_mm               64
width_mm                64
height_mm               64
seating_capacity        64
fuel_tank_capacity_l   113
dtype: int64
```

Remove missing rows

```
df = df.dropna()
```

```
print(df.isna().sum())
```

```
make                   0
model                  0
price_usd              0
year                   0
kilometer              0
fuel_type              0
transmission           0
color                  0
owner                  0
seller_type            0
engine                 0
max_power              0
max_torque             0
drivetrain             0
length_mm              0
width_mm               0
height_mm              0
seating_capacity       0
fuel_tank_capacity_l   0
dtype: int64
```

Drop Duplicate Information

```
duplicates_data = df.duplicated()
num_dup = duplicates_data.sum()
print("Numbers of Duplicate: ", num_dup)
```

```
Numbers of Duplicate:  4
```

```
df.drop_duplicates(inplace=True)
```

Change Datatype (From str to int)

```
# def extract_numerical_values(s):
#     values = s.split('@')[0].strip().split()
#     return float(values[0]), int(values[1])

df[['max_power_bhp','max_power_rpm']] = df['max_power'].str.split('@', expand = True)
# df['max_torque_Nm'], df['max_torque_rpm'] = zip(df['max_torque'].apply(extract_numerical_values))

# df.drop(['max_power', 'max_torque'], axis=1, inplace=True)

# print(df)
df
```

| | make | model | price_usd | year | kilometer | fuel_type | transmission | color | o |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Honda | Amaze 1.2 VX i-VTEC | 6060 | 2017 | 87150 | Petrol | Manual | Grey | |
| **1** | Maruti Suzuki | Swift DZire VDI | 5400 | 2014 | 75000 | Diesel | Manual | White | Se |
| **2** | Hyundai | i10 Magna 1.2 Kappa2 | 2640 | 2011 | 67000 | Petrol | Manual | Maroon | |
| **3** | Toyota | Glanza G | 9588 | 2019 | 37500 | Petrol | Manual | Red | |
| **4** | Toyota | Innova 2.4 VX 7 STR [2016-2020] | 23400 | 2018 | 69000 | Diesel | Manual | Grey | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2053** | Maruti Suzuki | Ritz Vxi (ABS) BS-IV | 2940 | 2014 | 79000 | Petrol | Manual | White | Se |
| **2054** | Mahindra | XUV500 W8 [2015-2017] | 10200 | 2016 | 90300 | Diesel | Manual | White | |
| **2055** | Hyundai | Eon D-Lite + | 3300 | 2014 | 83000 | Petrol | Manual | White | Se |
| **2056** | Ford | Figo Duratec Petrol ZXI 1.2 | 2880 | 2013 | 73000 | Petrol | Manual | Silver | |
| **2057** | BMW | 5-Series 520d Luxury Line [2017-2019] | 51480 | 2018 | 60474 | Diesel | Automatic | White | |

1870 rows × 21 columns

```
df[['max_torque_bhp','max_torque_rpm']] = df['max_torque'].str.split('@', expand = True)
df
```

| | make | model | price_usd | year | kilometer | fuel_type | transmission | color | own |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Honda | Amaze 1.2 VX i-VTEC | 6060 | 2017 | 87150 | Petrol | Manual | Grey | Fi |
| **1** | Maruti Suzuki | Swift DZire VDI | 5400 | 2014 | 75000 | Diesel | Manual | White | Seco |
| **2** | Hyundai | i10 Magna 1.2 Kappa2 | 2640 | 2011 | 67000 | Petrol | Manual | Maroon | Fi |
| **3** | Toyota | Glanza G | 9588 | 2019 | 37500 | Petrol | Manual | Red | Fi |
| **4** | Toyota | Innova 2.4 VX 7 STR [2016-2020] | 23400 | 2018 | 69000 | Diesel | Manual | Grey | Fi |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2053** | Maruti Suzuki | Ritz Vxi (ABS) BS-IV | 2940 | 2014 | 79000 | Petrol | Manual | White | Seco |
| **2054** | Mahindra | XUV500 W8 [2015-2017] | 10200 | 2016 | 90300 | Diesel | Manual | White | Fi |
| **2055** | Hyundai | Eon D-Lite + | 3300 | 2014 | 83000 | Petrol | Manual | White | Seco |
| **2056** | Ford | Figo Duratec Petrol ZXI 1.2 | 2880 | 2013 | 73000 | Petrol | Manual | Silver | Fi |
| **2057** | BMW | 5-Series 520d Luxury Line [2017-2019] | 51480 | 2018 | 60474 | Diesel | Automatic | White | Fi |

370 rows × 23 columns

Feature Selection

```
df.drop(columns=['model'], inplace=True)
```

```
df
```

| 'etrain | length_mm | width_mm | height_mm | seating_capacity | fuel_tank_capacity_l | max_power |
|---|---|---|---|---|---|---|
| FWD | 3990.0 | 1680.0 | 1505.0 | 5.0 | 35.0 | 8 |
| FWD | 3995.0 | 1695.0 | 1555.0 | 5.0 | 42.0 | 7 |
| FWD | 3585.0 | 1595.0 | 1550.0 | 5.0 | 35.0 | 7 |
| FWD | 3995.0 | 1745.0 | 1510.0 | 5.0 | 37.0 | 8 |
| RWD | 4735.0 | 1830.0 | 1795.0 | 7.0 | 55.0 | 14 |
| ... | ... | ... | ... | ... | ... | |
| FWD | 3775.0 | 1680.0 | 1620.0 | 5.0 | 43.0 | 8 |
| FWD | 4585.0 | 1890.0 | 1785.0 | 7.0 | 70.0 | 13 |
| FWD | 3495.0 | 1550.0 | 1500.0 | 5.0 | 32.0 | 5 |
| FWD | 3795.0 | 1680.0 | 1427.0 | 5.0 | 45.0 | 7 |
| RWD | 4936.0 | 1868.0 | 1479.0 | 5.0 | 65.0 | 18 |

```
# Convert 'engine' column to numeric
df['engine'] = df['engine'].str.extract(r'(\d+)')
df['engine'] = pd.to_numeric(df['engine'])

# Convert 'max_power_bhp' column to numeric
df['max_power_bhp'] = df['max_power_bhp'].str.extract(r'(\d+)')
df['max_power_bhp'] = pd.to_numeric(df['max_power_bhp'])

# Convert 'max_power_rpm' column to numeric
df['max_power_rpm'] = df['max_power_rpm'].str.extract(r'(\d+)')
df['max_power_rpm'] = pd.to_numeric(df['max_power_rpm'])

# Convert 'max_torque_bhp' column to numeric
df['max_torque_bhp'] = df['max_torque_bhp'].str.extract(r'(\d+)')
df['max_torque_bhp'] = pd.to_numeric(df['max_torque_bhp'])

# Convert 'max_torque_rpm' column to numeric
df['max_torque_rpm'] = df['max_torque_rpm'].str.extract(r'(\d+)')
df['max_torque_rpm'] = pd.to_numeric(df['max_torque_rpm'])
```

```
df
```

| | make | price_usd | year | kilometer | fuel_type | transmission | color | owner | sel |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Honda | 6060 | 2017 | 87150 | Petrol | Manual | Grey | First | |
| **1** | Maruti Suzuki | 5400 | 2014 | 75000 | Diesel | Manual | White | Second | |
| **2** | Hyundai | 2640 | 2011 | 67000 | Petrol | Manual | Maroon | First | |
| **3** | Toyota | 9588 | 2019 | 37500 | Petrol | Manual | Red | First | |
| **4** | Toyota | 23400 | 2018 | 69000 | Diesel | Manual | Grey | First | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **2053** | Maruti Suzuki | 2940 | 2014 | 79000 | Petrol | Manual | White | Second | |
| **2054** | Mahindra | 10200 | 2016 | 90300 | Diesel | Manual | White | First | |
| **2055** | Hyundai | 3300 | 2014 | 83000 | Petrol | Manual | White | Second | |
| **2056** | Ford | 2880 | 2013 | 73000 | Petrol | Manual | Silver | First | |
| **2057** | BMW | 51480 | 2018 | 60474 | Diesel | Automatic | White | First | |

1870 rows × 22 columns

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Assuming df_renovate is a DataFrame containing your data
df_renovate = pd.read_csv('used_car_price.csv')  # Replace 'your_data.csv' with your actual data file

X = df_renovate['kilometer'].values
y = df_renovate['price_usd'].values

lr = LinearRegression()
lr.fit(X.reshape(-1, 1), y)

y_pred_lr = lr.predict(X.reshape(-1, 1))

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred_lr, color='red', label='Linear Regression')
plt.title('Linear Regression: Kilometer vs Price')
plt.xlabel('Kilometer')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```


```
r_squared = lr.score(X.reshape(-1, 1), y)
print(f"R-squared value: {r_squared}")
```

```
R-squared value: 0.022748173421355933
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer

X = df_renovate[['kilometer', 'engine', 'fuel_tank_capacity_l']].values
y = df_renovate['price_usd'].values

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X_imputed)

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

poly_lr = LinearRegression()
poly_lr.fit(X_train, y_train)

y_pred_train = poly_lr.predict(X_train)
y_pred_test = poly_lr.predict(X_test)

train_rmse = mean_squared_error(y_train, y_pred_train, squared=False)
test_rmse = mean_squared_error(y_test, y_pred_test, squared=False)
r2t = r2_score(y_train, y_pred_train)
r2s = r2_score(y_test, y_pred_test)

print(f"Train RMSE: {train_rmse}")
print(f"Test RMSE: {test_rmse}")
print(f"Train r2: {r2t}")
print(f"Test r2: {r2s}")

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_test, color='blue', label='Test Data')
plt.plot([0, 20000], [0, 20000], color='red', linestyle='--', linewidth=2, label='Perfect Fit')
plt.title('Actual vs Predicted Prices (Test Data)')
plt.xlabel('Actual Price (USD)')
plt.ylabel('Predicted Price (USD)')
plt.xlim(0, 20000)  # Limit x-axis to 20000
plt.ylim(0, 20000)  # Limit y-axis to 20000
plt.legend()
plt.grid(True)
plt.show()
```

```
Train RMSE: 16864.65208793222
Test RMSE: 23321.90500580849
Train r2: 0.6453558036142135
```