Name: Parwinder Singh
Batch Code: LISUM21
Submission Date: July 19, 2023
Submitted to: GitHub

Step 1: Selected EV data

In [5]:
```
print(clean_ev_data)
```

```
            Brand                              Model  AccelSec  TopSpeed_KmH  \
0           Tesla     Model 3 Long Range Dual Motor       4.6           233
1      Volkswagen                            ID.3 Pure      10.0           160
2        Polestar                                  2        4.7           210
3             BMW                               iX3         6.8           180
4           Honda                                 e         9.5           145
..            ...                               ...       ...           ...
98         Nissan                         Ariya 63kWh       7.5           160
99           Audi     e-tron S Sportback 55 quattro       4.5           210
100        Nissan               Ariya e-4ORCE 63kWh       5.9           200
101        Nissan   Ariya e-4ORCE 87kWh Performance       5.1           200
102         Byton                  M-Byte 95 kWh 2WD       7.5           190

     Range_Km  Efficiency_WhKm FastCharge_KmH RapidCharge PowerTrain  \
0         450              161            940         Yes        AWD
1         270              167            250         Yes        RWD
2         400              181            620         Yes        AWD
3         360              206            560         Yes        RWD
4         170              168            190         Yes        RWD
```

In [6]:
```
print(norm_ev_data)
```

```
            Brand                              Model     Accel  TopSpeed   Range  \
0           Tesla     Model 3 Long Range Dual Motor   4.6 sec  233 km/h  450 km
1      Volkswagen                            ID.3 Pure  10.0 sec  160 km/h  270 km
2        Polestar                                  2   4.7 sec  210 km/h  400 km
3             BMW                               iX3    6.8 sec  180 km/h  360 km
4           Honda                                 e    9.5 sec  145 km/h  170 km
..            ...                               ...       ...       ...     ...
98         Nissan                         Ariya 63kWh   7.5 sec  160 km/h  330 km
99           Audi     e-tron S Sportback 55 quattro   4.5 sec  210 km/h  335 km
100        Nissan               Ariya e-4ORCE 63kWh   5.9 sec  200 km/h  325 km
101        Nissan   Ariya e-4ORCE 87kWh Performance   5.1 sec  200 km/h  375 km
102         Byton                  M-Byte 95 kWh 2WD   7.5 sec  190 km/h  400 km

    Efficiency FastCharge              RapidCharge          PowerTrain  \
0    161 Wh/km   940 km/h  Rapid charging possible    All Wheel Drive
1    167 Wh/km   250 km/h  Rapid charging possible   Rear Wheel Drive
2    181 Wh/km   620 km/h  Rapid charging possible    All Wheel Drive
3    206 Wh/km   560 km/h  Rapid charging possible   Rear Wheel Drive
4    168 Wh/km   190 km/h  Rapid charging possible   Rear Wheel Drive
..         ...        ...                       ...                ...
98   191 Wh/km   440 km/h  Rapid charging possible  Front Wheel Drive
99   258 Wh/km   540 km/h  Rapid charging possible    All Wheel Drive
100  194 Wh/km   440 km/h  Rapid charging possible    All Wheel Drive
101  232 Wh/km   450 km/h  Rapid charging possible    All Wheel Drive
102  238 Wh/km   480 km/h  Rapid charging possible    All Wheel Drive

       PlugType  BodyStyle Segment  Seats  PriceEuro
0    Type 2 CCS      Sedan       D      5      55480
1    Type 2 CCS  Hatchback       C      5      30000
2    Type 2 CCS   Liftback       D      5      56440
3    Type 2 CCS        SUV       D      5      68040
4    Type 2 CCS  Hatchback       B      4      32997
..          ...        ...     ...    ...        ...
98   Type 2 CCS  Hatchback       C      5      45000
99   Type 2 CCS        SUV       E      5      96050
100  Type 2 CCS  Hatchback       C      5      50000
101  Type 2 CCS  Hatchback       C      5      65000
102  Type 2 CCS        SUV       E      5      62000

[103 rows x 14 columns]
```

Step 2: Saved the Gradient Boosting Regression Model

Gradient Boosting Regression Model

In [11]:
```
# Selected features and target variables to build model around
X = norm_ev_data[['TopSpeed', 'Range']]
y = norm_ev_data['PriceEuro']
```

In [12]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5, random_state = 42)
```

In [13]:
```
# Created and trained the Gradient Boosting Regression model
gbr_model = GradientBoostingRegressor()

# Trained the model using the training data
gbr_model.fit(X_train, y_train)
```

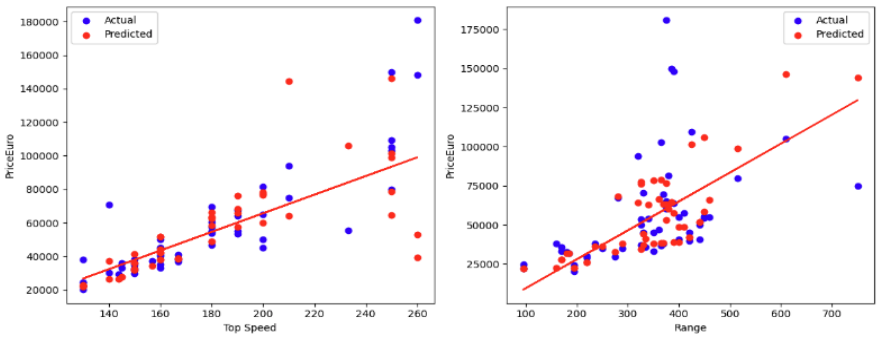
Out[13]:
```
GradientBoostingRegressor
GradientBoostingRegressor()
```

In [14]:
```
# Saved the trained model to a file for Flask deployment
with open('gbr_model.pkl', 'wb') as file:
    pickle.dump(gbr_model, file)
```

In [15]:
```
# Made predictions on the test set
y_pred = gbr_model.predict(X_test)

# Calculated the mean squared error and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print('Mean Squared Error:', mse)
print('R-squared Score:', r2)
```

```
Mean Squared Error: 998166209.4394575
R-squared Score: 0.09532714133017783
```

In [16]:
```
plt.figure(figsize=(12, 5))

# Scatter plot for TopSpeed vs PriceEuro
plt.subplot(1, 2, 1)
plt.scatter(X_test['TopSpeed'], y_test, color='blue', label='Actual')
plt.scatter(X_test['TopSpeed'], y_pred, color='red', label='Predicted')
plt.xlabel('Top Speed')
plt.ylabel('PriceEuro')
plt.legend()

# Added the trend line for TopSpeed vs PriceEuro
trend_line_top_speed = np.polyfit(X_test['TopSpeed'], y_pred, 1)
plt.plot(X_test['TopSpeed'], np.polyval(trend_line_top_speed, X_test['TopSpeed']), color='red')

# Scatter plot for Range vs PriceEuro
plt.subplot(1, 2, 2)
plt.scatter(X_test['Range'], y_test, color='blue', label='Actual')
plt.scatter(X_test['Range'], y_pred, color='red', label='Predicted')
plt.xlabel('Range')
plt.ylabel('PriceEuro')
plt.legend()

# Added the trend line for Range vs PriceEuro
trend_line_range = np.polyfit(X_test['Range'], y_pred, 1)
plt.plot(X_test['Range'], np.polyval(trend_line_range, X_test['Range']), color='red')

plt.tight_layout()
plt.show()
```



Deployed the model on to Flask app called “Par_App”

```
from flask import Flask, request, jsonify

app = Flask('Par_App')

# Load the trained model
with open('gbr_model.pkl', 'rb') as file:
    gbr_model = pickle.load(file)

# Define an API endpoint for predictions
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    top_speed = data['TopSpeed']
    range_km = data['Range']
    prediction = gbr_model.predict([[top_speed, range_km]])
    return jsonify({'prediction': float(prediction[0])})

if __name__ == '__main__':
    app.run(debug = True, port = 9000)
```

```
 * Serving Flask app 'Par_App'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a produc
tion WSGI server instead.
 * Running on http://127.0.0.1:9000
```