Used Automobile Price Prediction

Mini Project 2 - Subashanan Nair

Problem Statement

The problem at hand is to predict the price of a used car based on various features such as year of manufacture, km driven, fuel type, etc.

Kaggle Dataset

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Car details v3.csv')
df.head()
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

Machine Learning Models used

- Linear Regression
- Random Forest
- XGBoost
- Lasso Regression

Training the model with test data

```
[137]  1 for name, model in models:
       2     model.fit(X_train, y_train)
       3     y_pred = model.predict(X_test)
       4     mse = mean_squared_error(y_test, y_pred)
       5     rmse = np.sqrt(mse)
       6     r2 = r2_score(y_test, y_pred)
       7     print(f"{name}: RMSE={rmse:.2f}, R2={r2:.2f}")
       8
```

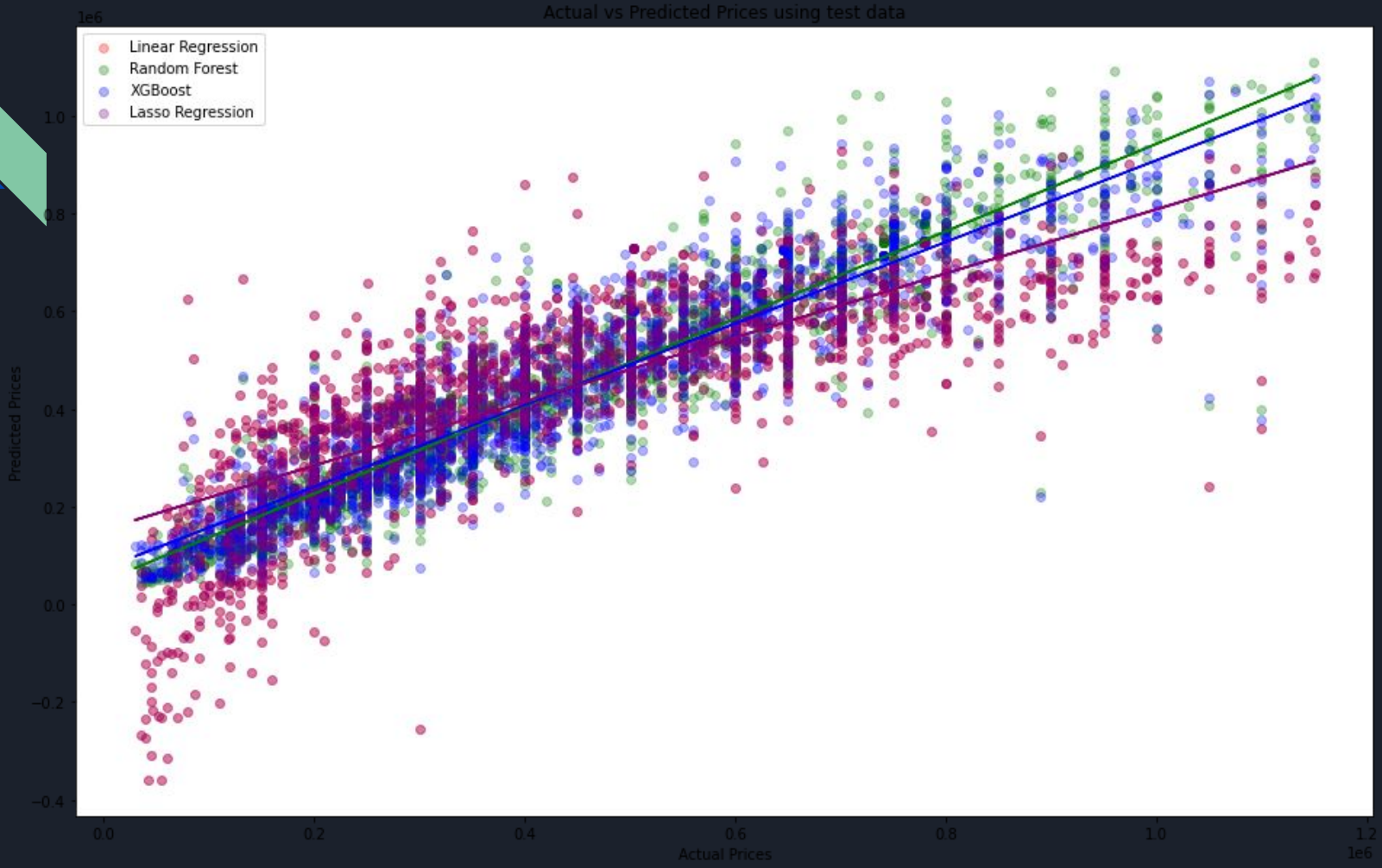
```
Linear Regression: RMSE=144750.07, R2=0.65
Random Forest: RMSE=79685.97, R2=0.89
[07:15:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBoost: RMSE=92232.28, R2=0.86
Lasso Regression: RMSE=144750.25, R2=0.65
```

The Linear Regression model has a RMSE of 144750.07 and R2 score of 0.65, indicating that the model is not very accurate in predicting the target variable.

The Random Forest model has a RMSE of 79685.97 and R2 score of 0.89, indicating that the model is highly accurate in predicting the target variable.

The XGBoost model has a RMSE of 92232.28 and R2 score of 0.86, indicating that it is a highly accurate model for predicting the target variable, compared to the Linear Regression model.

The Lasso Regression model has a RMSE of 144750.25 and R2 score of 0.65, which is similar to the Linear Regression model in terms of accuracy for predicting the target variable.


Actual vs Predicted Prices using test data
1e6
Linear Regression
Random Forest
XGBoost
Lasso Regression
Predicted Prices
Actual Prices
1e6
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0
1.2

Training the model with train data

```
[161]  1 for name, model in models:
       2   model.fit(X_train, y_train)
       3   y_pred = model.predict(X_train)
       4   mse = mean_squared_error(y_train, y_pred)
       5   rmse = np.sqrt(mse)
       6   r2 = r2_score(y_train, y_pred)
       7   print(f"{name}: RMSE={rmse:.2f}, R2={r2:.2f}")
```

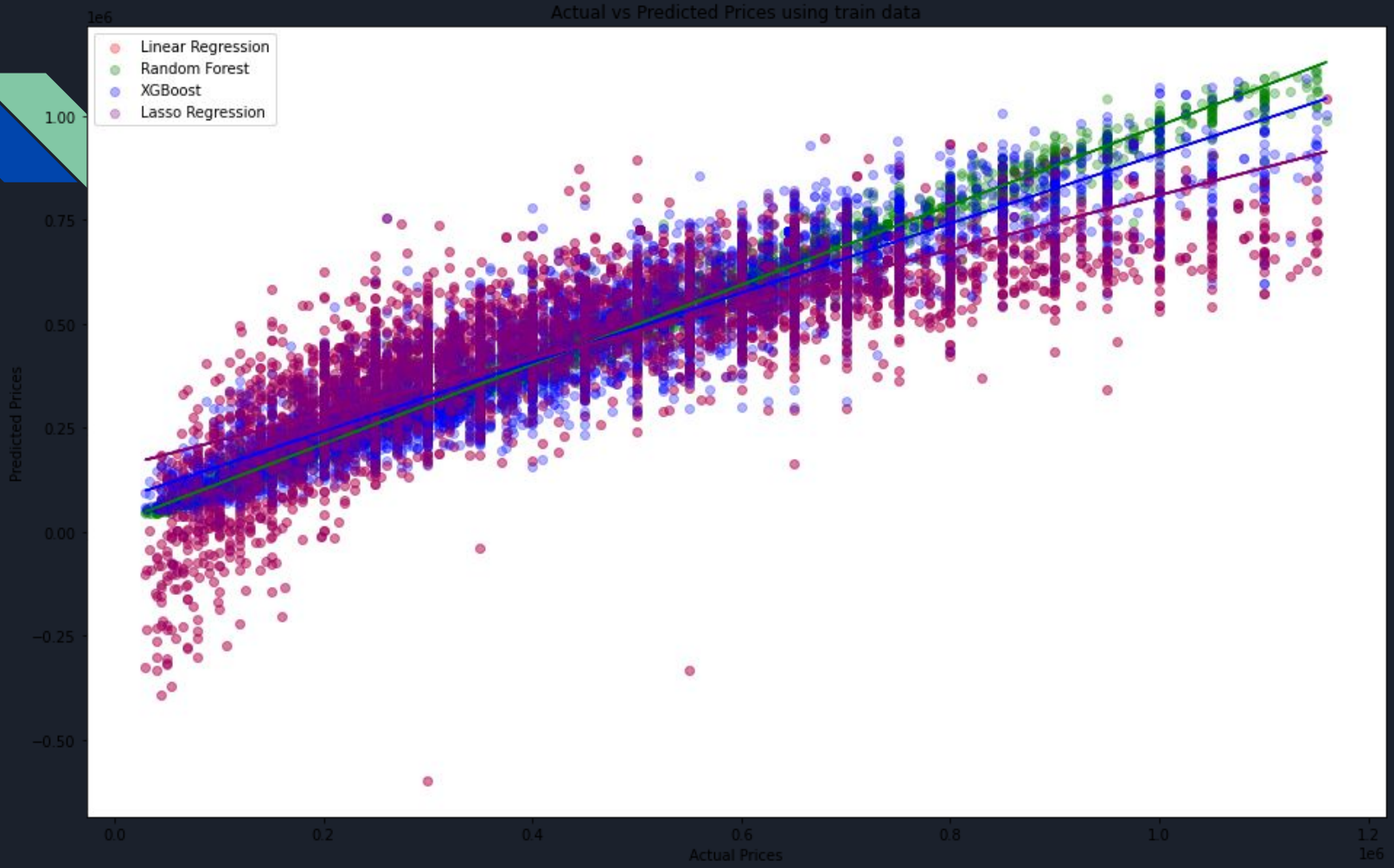
```
Linear Regression: RMSE=143619.53, R2=0.66
Random Forest: RMSE=33524.23, R2=0.98
[08:46:33] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBoost: RMSE=89372.12, R2=0.87
Lasso Regression: RMSE=143619.53, R2=0.66
```

The Linear Regression model has a RMSE of 143619.53 and R2 score of 0.66, indicating that the model has a moderate accuracy in predicting the dependent variable.

The Random Forest model has a RMSE of 33524.23 and R2 score of 0.98, indicating that the model has a very high accuracy in predicting the dependent variable.

The XGBoost model has a RMSE of 89372.12 and R2 score of 0.87, indicating that the model is good at predicting the dependent variable but not as accurate as the Random Forest model.

The Lasso Regression model has a RMSE of 143619.53 and R2 score of 0.66, which is similar to the Linear Regression model in terms of accuracy.


Actual vs Predicted Prices using train data
1e6
Linear Regression
Random Forest
XGBoost
Lasso Regression
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Predicted Prices
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
Actual Prices

HyperParameter Tuning (RandomForest)

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best hyperparameters: {'n_estimators': 500, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': None}
RMSE: 77736.08259917068
R2 score: 0.8984801752097382
         Feature  Importance
0           year    0.340541
7         torque    0.225986
8         engine    0.153058
1      km_driven    0.084348
9        mileage    0.080681
5    owner_float    0.039267
6          seats    0.026931
2           fuel    0.025672
4   transmission    0.013833
3    seller_type    0.009683
```

Based on the output from the hyperparameter tuning, the best hyperparameters for the Random Forest model are n_estimators = 500, min_samples_split = 2, min_samples_leaf = 1, max_features = 'log2', and max_depth = None. With these hyperparameters, the Random Forest model has a Root Mean Squared Error (RMSE) of 77736.08 and an R-squared score of 0.898, indicating that the model has a good fit on the data.

Conclusion

The feature importances show that the year of the vehicle is the most important feature in determining the price, followed by torque, engine, and km_driven. The other features, such as owner_float, seats, fuel, transmission, seller_type, and mileage, have relatively lower importance.

Based on the results, the Random Forest Regression model performed better than the Linear Regression model, with a lower RMSE and higher R-squared score. The model can be used to predict the prices of used cars based on their features, with a high degree of accuracy.