MSDS 692 Data Science Practicum I:

Final Report: Predicting the Price of Used Cars

Babawale Olatunji

Regis University

Authors Note

This Final Report was prepared for MSDS 692 taught by Prof. Hart Douglas

Predicting the Price of Used Cars

1. Introduction

Predicting the price of used cars is a challenging task. This is due to several features that should be examined for accurate prediction. Potential used car buyers do not know if they are paying too much for the selling price of used cars. It is a difficult task for potential used car buyers to keep track of all the interesting used cars available on the automobile market at any given time. Furthermore, they are not even cognizant of the comparable used cars with similar car features that are available on the automobile market at any given period.

The goal of this data science practicum project is to develop a machine learning model that can be leveraged in predicting price of used cars. This model can further be used in predicting the price value for a single car model, top selling car models, features importance on car price value, and averaged time elapsed before selling a car.

2. Data

2.1 Presentation of the data

This project utilized Kaggle dataset located at https://www.kaggle.com/orgesleka/used-cars-database to achieve its objectives. The data contains offerings of used cars in Germany. The content of the original dataset is in German. Google Translator was used to make the necessary translations to English. The data was scraped or collected with Python Scrapy from Ebay-Kleinanzeigen and have been crawled between 03-05-2016 and 04-07-2016. The data contains over 370,000 used cars information, each characterized by the following 20 variables:

- dateCrawled: The date when this advertisement was first crawled, all field-values were obtained on this date;
- name: "name" of the car;

- seller: seller type – private or dealer;
- offerType: Offer Type - offer or request;
- price: the price in Euro on the advertisement to sell the car;
- abtest: abtest category - test or control;
- vehicleType: vehicle body type - limousine, small car, station wagon, bus, cabrio, coupe, suv, other;
- yearOfRegistration: At what year the car was first registered - the age of the car;
- Transmission: Transmission Type - manual or automatic;
- powerPS: Car Engine Power in PS;
- model: car model;
- kilometer: car mileage in kilometer;
- monthOfRegistration: the month of the year the car was first registered;
- fuelType: Fuel Type - gas, diesel, autogas, compressed natural gas, hybrid, other, or electric;
- brand: car brand;
- notRepairedDamage: Unrepaired Damage - yes or no;
- dateCreated: The date the ad was created on Ebay-Kleinanzeigen;
- nrOfPictures: number of pictures in the ad;
- postalCode: car seller postal code;
- lastSeen: when the crawler saw this ad last online.

The data is consisting of 14 string variables and 6 numerical variables.

2.2 Data Preparation

There are missing values identified as NaNs/Null and zeros (0.0) in the dataset used for this project. Missing values if not dealt with would result in bias resulting from the differences between missing and complete data. I dealt with missing values in the dataset by dropping the rows and columns that contain them. Figure 1 and Figure 2 depict the summary count of missing values by column before and after data cleaning.

```
Date_Crawled                     0
Car_Name                         0
Car_Seller_Type                  0
Offer_Type                       0
Car_Price                    10772
Abtest_Type                      0
Vechicle_Type                37862
Year_of_Car_Registration         0
Car_Transmission_Type        20203
Car_Engine_Power_PS          40812
Car_Model                    20481
Car_Mileage_Kilometer            0
Month_of_Car_Registration    37670
Fuel_Type                    33379
Car_Brand                        0
UnRepaired_Damage            72053
Date_Created                     0
No_of_Pictures              371522
Seller_Postal_Code               0
Date_LastSeen_Online             0
dtype: int64
```

Figure 1: Summary count of missing values by column before data cleaning

```
autos_cleaned_df.isnull().sum()
```

```
Car_Price                    0
Vechicle_Type                0
Year_of_Car_Registration     0
Car_Transmission_Type        0
Car_Engine_Power_PS          0
Car_Model                    0
Car_Mileage_Kilometer        0
Month_of_Car_Registration    0
Fuel_Type                    0
Car_Brand                    0
UnRepaired_Damage            0
Date_Created                 0
Seller_Postal_Code           0
Date_LastSeen_Online         0
dtype: int64
```

Figure 2: Summary count of missing values by column after data cleaning

Further examination of the dataset revealed the dataset contains Outliers. These are values that are distant from other observations in the column. Interquartile range (IQR) from the summary statistics first quartile and third quartile was used to detect the Outliers from the relevant columns. Outlier values were removed from the data. Outliers would have a positive or negative effect on the correlation of the data if not handled correctly. Figure 3 highlights the summary of the dataframe including the data type of each column after the data cleaning.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 242202 entries, 3 to 371539
Data columns (total 14 columns):
Car_Price                   242202 non-null float64
Vechicle_Type               242202 non-null object
Year_of_Car_Registration    242202 non-null float64
Car_Transmission_Type       242202 non-null object
Car_Engine_Power_PS         242202 non-null float64
Car_Model                   242202 non-null object
Car_Mileage_Kilometer       242202 non-null object
Month_of_Car_Registration   242202 non-null float64
Fuel_Type                   242202 non-null object
Car_Brand                   242202 non-null object
UnRepaired_Damage           242202 non-null object
Date_Created                242202 non-null datetime64[ns]
Seller_Postal_Code          242202 non-null float64
Date_LastSeen_Online        242202 non-null datetime64[ns]
dtypes: datetime64[ns](2), float64(5), object(7)
memory usage: 27.7+ MB
```

Figure 3: Summary of the dataframe after data cleaning

2.3 Exploratory Data Analysis (EDA)

Summary Statistics of both Numerical and Non-Numerical Attributes were used to highlight the overall view of the cleaned dataset. Table 1 depicts the Summary Statistics of Non-Numerical Attributes.

Table 1: Summary Statistics of Non-Numerical Attributes

	Vechicle_Type	Car_Transmission_Type	Car_Model	Fuel_Type	Car_Brand	UnRepaired_Damage
count	242202	242202	242202	242202	242202	242202
unique	8	2	250	7	39	2
top	limousine	manual	golf	gas	volkswagen	no
freq	71451	184818	19618	155144	50437	219292

The above Table 1 shows limousine, manual, golf, gas and Volkswagen are the top Vehicle Type, Car Transmission Type, Car Model, Fuel Type and Car Brand respectively. Furthermore, Table 2 below depicts the Summary Statistics of Numerical Attributes highlighting the mean, standard deviation, first, second and third quartiles, minimum and maximum descriptive statistics for each of the numerical variables.

Table 2: Summary Statistics of Numerical Attributes

	Car_Price	Year_of_Car_Registration	Car_Engine_Power_PS	Car_Mileage_Kilometer	Month_of_Car_Registration
count	242202.000000	242202.000000	242202.000000	242202.000000	242202.000000
mean	6877.679239	2003.599326	129.682269	123474.702934	6.367309
std	8073.523840	6.338326	62.153342	39931.509539	3.350762
min	1.000000	1951.000000	1.000000	5000.000000	1.000000
25%	1700.000000	2000.000000	86.000000	100000.000000	3.000000
50%	4000.000000	2004.000000	116.000000	150000.000000	6.000000
75%	8990.000000	2008.000000	160.000000	150000.000000	9.000000
max	99999.000000	2016.000000	999.000000	150000.000000	12.000000

Histograms of numerical variables were created to visualize the distribution of the data for the numerical features. Figure 4 showcases the histogram for each of the numerical variable. Each histogram below highlights if the distribution of data for each of the numerical variable is symmetric, left-skewed or right-skewed. It is worth noting that Age of Car in months variable was calculated from the dataset Year of Registration and Month of Registration variables.

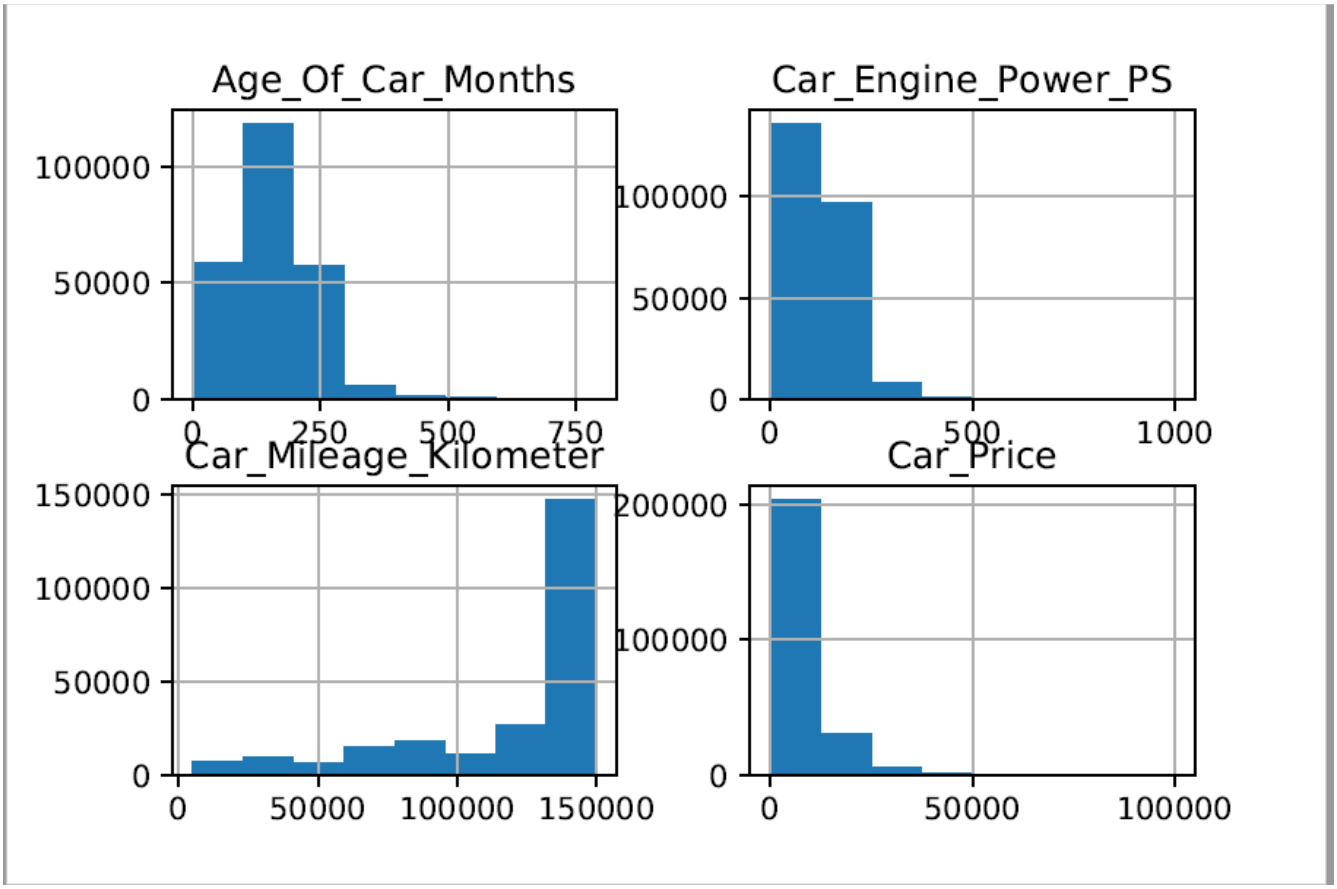

Figure 4: Histograms of Numerical Attributes

Pairs Plots is another effective tool to leverage in Exploratory Data Analysis. Pairs Plots were created for this project to visualize both the distribution of single variables including Car Price, numerical and non-numerical variables and the relationships between Car Price and each of the numerical and non-numerical variables. Figure 5 depicts the

Pairs Plots of Car Price and Numerical Attributes and Figure 6 showcases the Pairs Plots of Car Price and Non-Numerical Attributes.

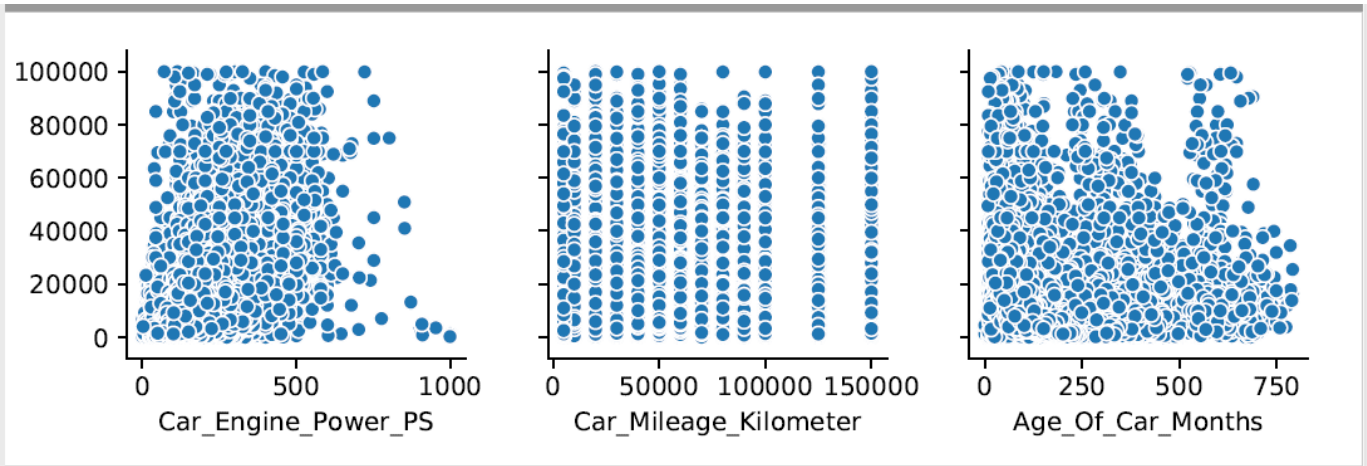

Figure 5: Pairs Plots of Car Price and Numerical Attributes

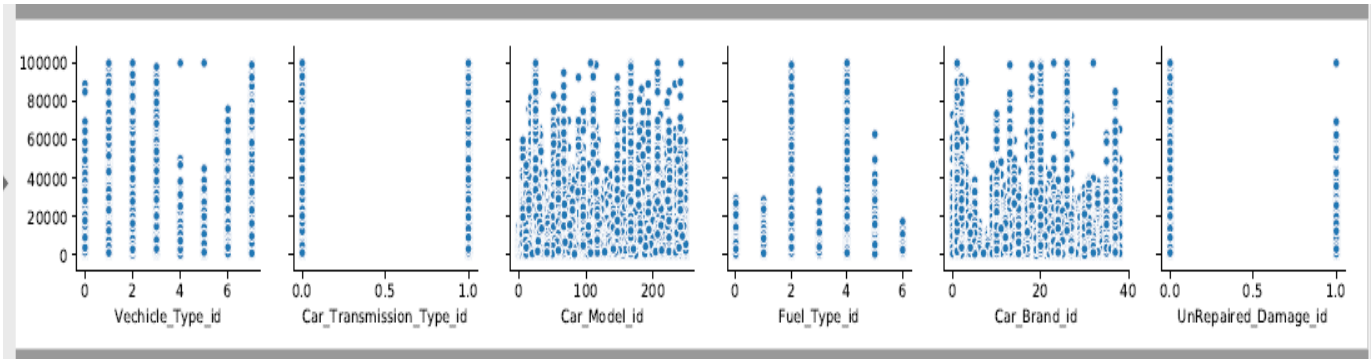

Figure 6: Pairs Plots of Car Price and Non-Numerical Attributes

To find out the correlation coefficient, the correlation matrix was developed to visualize how each attribute of the project dataset is compared to the other factors. Specifically, the correlation between the Car Price and each of the other attributes. Figure 7 depicts the correlation matrix of the dataset attributes. The correlation results show a positive correlation of (0.59) between Car Price and Car Engine Power. That means an

increase in Car Engine Power would have an increase in the Car Price. Conversely, there is negative correlation between Car Mileage and Car Price. Also, there is a negative correlation between Age of Car and Car Price.

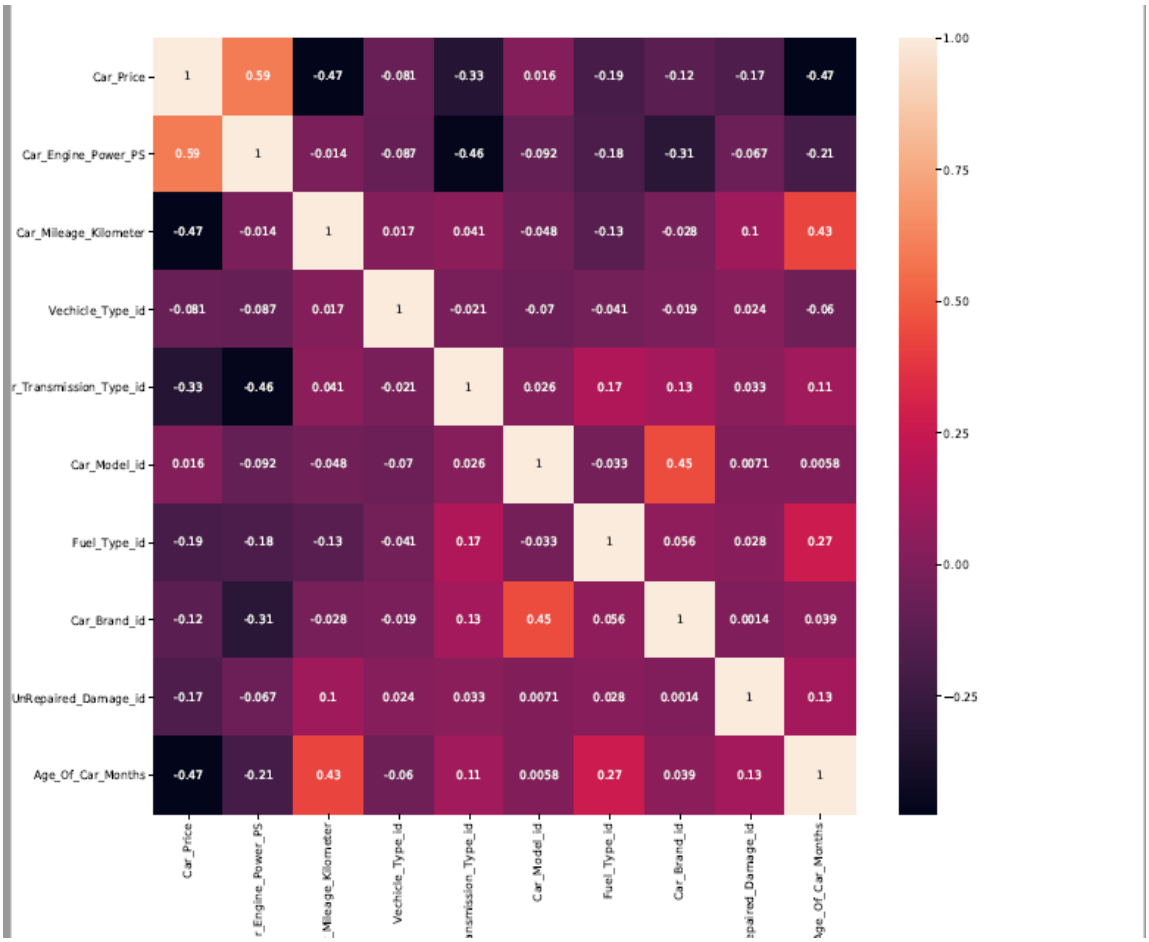

Figure 7: Correlation Matrix of the Attributes

3. Feature Selection

The performance of a machine learning model is greatly affected by feature selection. It is worth noting that model performance can be negatively affected by irrelevant or partially pertinent features. The process of manually or automatically selecting those features or input that contribute the most to the target or prediction variable is called Feature

Selection. It minimizes overfitting, fosters prediction accuracy and minimizes training time.

To determine the important features for the output (Car Price) variable, univariate selection process was leveraged in this project to identify relevant features (independent variables) that contribute the most to the target or output(dependent) variable.

```
# Display the 9 best features
print(featureScores.nlargest(9,'Score'))
```

```
                   Features         Score
1     Car_Mileage_Kilometer  8.683738e+08
8         Age_Of_Car_Months  3.773657e+06
0       Car_Engine_Power_PS  2.884651e+06
4              Car_Model_id  3.479805e+05
6              Car_Brand_id  9.487760e+04
7      UnRepaired_Damage_id  3.064678e+04
2           Vehicle_Type_id  1.088418e+04
5              Fuel_Type_id  8.850633e+03
3  Car_Transmission_Type_id  8.281381e+03
```

Figure 8: Univariate Feature Selection Result

The above result identifies Car_Mileage_Kilometer, Age_Of_Car_Months, and Car_Engine_Power_PS as the three most important features in predicting the price of used cars. This is also consistent with the correlation result above. This result can be utilized to determine if certain features should be included in the models to be constructed using this dataset.

4. Methods

4.1 Models

There are two types of supervised machine learning algorithms including Regression and Classification. Predicting the price of used cars is a regression problem. To predict a continuous value like the price of used cars, regression models are used. Using Python Scikit learn library, various kinds of regression models can be developed and implemented.

For this practicum project, I chose to utilize four different Python Scikit learn regression methods/algorithms for training and building four different regression models:

- Linear Regression: In this project, multiple linear regression is leveraged to model the relationship between multiple car features including Vehicle Type, Transmission Type, Car Mileage and car price as the response or outcome.
- Decision Tree Regression: This is the second model constructed in this project. It trains a model in a flowchart-like tree structure to predict data in the future.
- Random Forest Regression. This is the third model developed in this regression problem project. It is an ensemble method and it combines multiple decision trees in determining the final output.
- Ensemble Voting Regressor: This is the fourth model constructed in this project. It fits Linear Regressor, Decision Tree Regressor and Random Forest Regressor each on the training dataset. This ensemble voting regressor averages these three individual regressors' predictions to produce a final prediction.

4.2 Methods

To predict the price of used cars, I developed four machine learning models using Python Scikit learn library and based on the results obtained from each of the four models, the best model was selected.

To accomplish this, the dataset was separated into features (independent variables) or inputs (X) and target(y). In this project, the target (dependent variable) data is from the Car Price variable and features or input data are from the remaining attributes including Car Engine Power, Car Mileage, Vehicle Type and other dataset variables. Both Feature and Target data were then split into training and test datasets with a 70/30 ratio. The training data is the data leveraged for learning by the four models and the test data is the data utilized to measure the performance of the four models on unseen data.

4.2.1 Training Models

In this project, four training models were constructed using multiple linear regression, decision tree regression, random forest regression and ensemble voting regressor machine learning techniques. This was done by importing linear model, decision tree regressor, random forest regressor and voting regressor from Scikit learn library. This was followed with the creation of an instance of the class regression for each model (LinearRegression, DecisionTreeRegressor, RandomForestRegressor, VotingRegressors) which represent the regression model. Each regression model was leveraged to train the model on both features train (X_train) dataset and target(y_train) dataset. That means, fitting the training data to the regression model. This was followed with the generation of prediction score. It is worth noting that the score is the R squared metric indicating the goodness of fit of the predicted used car prices to the true used car prices. Each model was then provided with test data (X_test) and the predictions for the used car prices based on each model were generated. This was followed with the calculation of Mean Absolute Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE) by passing the true or observed car prices (y_test) and model predicted car prices to the respective scikit learn metrics mean error method. MAE is the mean of the absolute difference between values predicted by a model or an estimator (such as car price) and true or observed values and is a linear score. MSE is the averaged squared difference between values predicted by a model or an estimator and observed values. RMSE is the square root of the mean of the squared errors (MSE) and is the commonly used measure.

All the four models developed in this project have more than one input variable, hence it is not feasible to represent several variables in a plot. Instead, scatterplot was leveraged to plot each model predicted used car prices versus the observed used car prices.

4.2.2 Cross Validation

The train-test split based on one split is referred to as classical approach. However, more than one split is done in cross validation. In this project, I utilized 10 splits in cross validation. These splits are called Folds. Cross-Validation provides a better understanding of what is going on by training 10 different models. Cross-validation requires fitting the

same machine learning regression model multiple times leveraging different subsets of the data or different splits each time. Cross-Validation would reveal consistency in performance of the algorithm and data or reveal inconsistency for further investigation. Furthermore, cross-validation would ensure similar performance in production deployment of the model. Testing accuracy can vary greatly depending on which observation happen to be in the testing dataset. Therefore, it is important to use k-fold cross-validation to fix this problem.

All the four models were constructed again with cross validation using multiple linear regression, decision tree regression, random forest regression and ensemble voting regression machine learning techniques. The four models were run with the input data(X) and target data(y) ten times (Kfold times) and estimated the score or R squared metric each time. The mean function was used to find the average of these ten prediction scores and standard deviation function was utilized to calculate the standard deviation of the 10 prediction scores. This was followed with the calculation of Mean Absolute Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE) leveraging cross validation score method. Furthermore, scatterplot was utilized to visually highlights how R-squared values denote the scatter around the regression line and plot each model predicted used car prices versus the observed used car prices.

5 Results

5.1 Presentation of the results

The followings are the R-Squared, MAE, MSE, and RMSE model evaluation metrics generated by the four models based on a single 70:30 train to test split ratio:

R Squared Metric for Multiple Linear Regression

In [356]:
```
# Compute and Display R Squared Metric value
# for Multiple Linear Regression Model
from sklearn.metrics import r2_score
print("R_Squared of MLR Model is: %0.2f" % (r2_score(y_test, Linear_Regressor_pred)))
```

```
R_Squared of MLR Model is: 0.62
```

Multiple linear regression model based on single split of 70:30 ratio produced R Squared value of 0.62.

MAE, MSE and RMSE for Multiple Linear Regression

In [221]:
```
# Dislay MAE, MSE and RMSE for Linear Regression
print('MAE:', metrics.mean_absolute_error(y_test, Linear_Regressor_pred))
print('MSE:', metrics.mean_squared_error(y_test, Linear_Regressor_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,Linear_Regressor_pred)))
```

```
MAE: 3111.6998285111285
MSE: 24598349.6062194
RMSE: 4959.672328513185
```

The Multiple linear regression model based on a single split produced MAE metric value of 3111.70, MSE metric value of 24598349.60 and RMSE metric value of 4959.67.

Figure 9: Multiple Linear Regression Model Results

R Squared Metric for Decision Tree Regression Model

In [341]:
```
# Compute and Display R Squared Metric value
# for Decision Tree Regression Model
from sklearn.metrics import r2_score
print("R_Squared of Decision Tree Regression Model is: %0.2f" % (r2_score(y_test, Tree_Regressor_pred)))
```

```
R_Squared of Decision Tree Regression Model is: 0.82
```

Decision Tree regression model based on a single split with 70:30 ratio produced R Squared value of 0.82.

MAE, MSE and RMSE for Decision Tree Regression Model

In [280]:
```
# Dislay MAE, MSE and RMSE for Decision Tree Regression model
print('MAE:', metrics.mean_absolute_error(y_test, Tree_Regressor_pred))
print('MSE:', metrics.mean_squared_error(y_test, Tree_Regressor_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,Tree_Regressor_pred)))
```

```
MAE: 1520.8651744667957
MSE: 11697553.492513902
RMSE: 3420.1686350988457
```

The Decision Tree regression model based on a single split produced MAE metric value of 1520.87, MSE metric value of 11697553.49 and RMSE metric value of 3420.17.

Figure 10: Decision Tree Regression Model Results

R Squared Metric for Random Forest Regression Model

In [345]:
```
# Compute and Display R Squared Metric value
# for Random Forest Regression Model
from sklearn.metrics import r2_score
print("R_Squared for Random Forest Regression Model is: %0.2f" % (r2_score(y_test, Rd_forest_Regressor_pred)))
```

```
R_Squared for Random Forest Regression Model is: 0.89
```

Random Forest regression model based on single split with 70:30 ratio produced R Squared value of 0.89.

MAE, MSE and RMSE for Random Forest Regression Model

In [298]:
```
# Dislay MAE, MSE and RMSE for Random Forest Regressor
print('MAE:', metrics.mean_absolute_error(y_test, Rd_forest_Regressor_pred))
print('MSE:', metrics.mean_squared_error(y_test, Rd_forest_Regressor_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,Rd_forest_Regressor_pred)))
```

```
MAE: 1288.38336720473
MSE: 7154066.3456259845
RMSE: 2674.7086468671655
```

Random Forest regression model based on a single split produced MAE metric value of 1288.38, MSE metric value of 7154066.35 and RMSE metric value of 2674.71.

Figure 11: Random Forest Regression Model Results

R Squared Metric for Ensemble Voting Regressor Model

In [352]:
```
# Compute and Display R Squared Metric value
# for Ensemble Voting Regressor Model
from sklearn.metrics import r2_score
print("R_Squared of Ensemble Voting Regressor Model is: %0.2f" % (r2_score(y_test,ensemble_pred)))
```

```
R_Squared of Ensemble Voting Regressor Model is: 0.86
```

```
Ensemble Voting Regressor model based on a single split with 70:30 ratio produced R Squared value of 0.86.
```

MAE, MSE and RMSE for Ensemble Voting Regressor Model

In [316]:
```
# Dislay MAE, MSE and RMSE for ensemble model
print('MAE:', metrics.mean_absolute_error(y_test, ensemble_pred))
print('MSE:', metrics.mean_squared_error(y_test, ensemble_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,ensemble_pred)))
```

```
MAE: 1626.8086093107863
MSE: 8912661.772720661
RMSE: 2985.4081417321586
```

Ensemble Voting Regressor model based on a single split produced MAE metric value of 1626.81, MSE metric value of 8912661.77 and RMSE metric value of 2985.41.

Figure 12: Ensemble Voting Regressor Model Results

Table 3: Regression Models Results Comparison

Evaluation Metrics	Multiple Linear Regression Model	Decision Tree Regression Model	Random Forest Regression Model	Ensemble Voting Regressor Model
R-Squared	0.62	0.82	0.89	0.86
Mean Absolute Error (MAE)	3111.70	1520.87	1288.38	1626.81
Mean Square Error (MSE)	24598349.60	11697553.49	7154066.35	8912661.77
Root Mean Square Error (RMSE)	4959.67	3420.17	2674.71	2985.41

From the results in Table 3 above, Random Forest Regression model based on a single 70/30 train to test ratio split generated R-Squared metric of 0.89, Mean Absolute Error (MAE) of 1288.38, Mean Square Error (MSE) of 7154066.55 and Root Mean Square Error metric of 2674.71. Random Forest model produced the best metrics among the four models developed, hence it is best model selected based on these four-evaluation metrics for predicting the price of used cars.

The followings are the Scatterplots depicting the four models predicted car prices versus the true car prices:

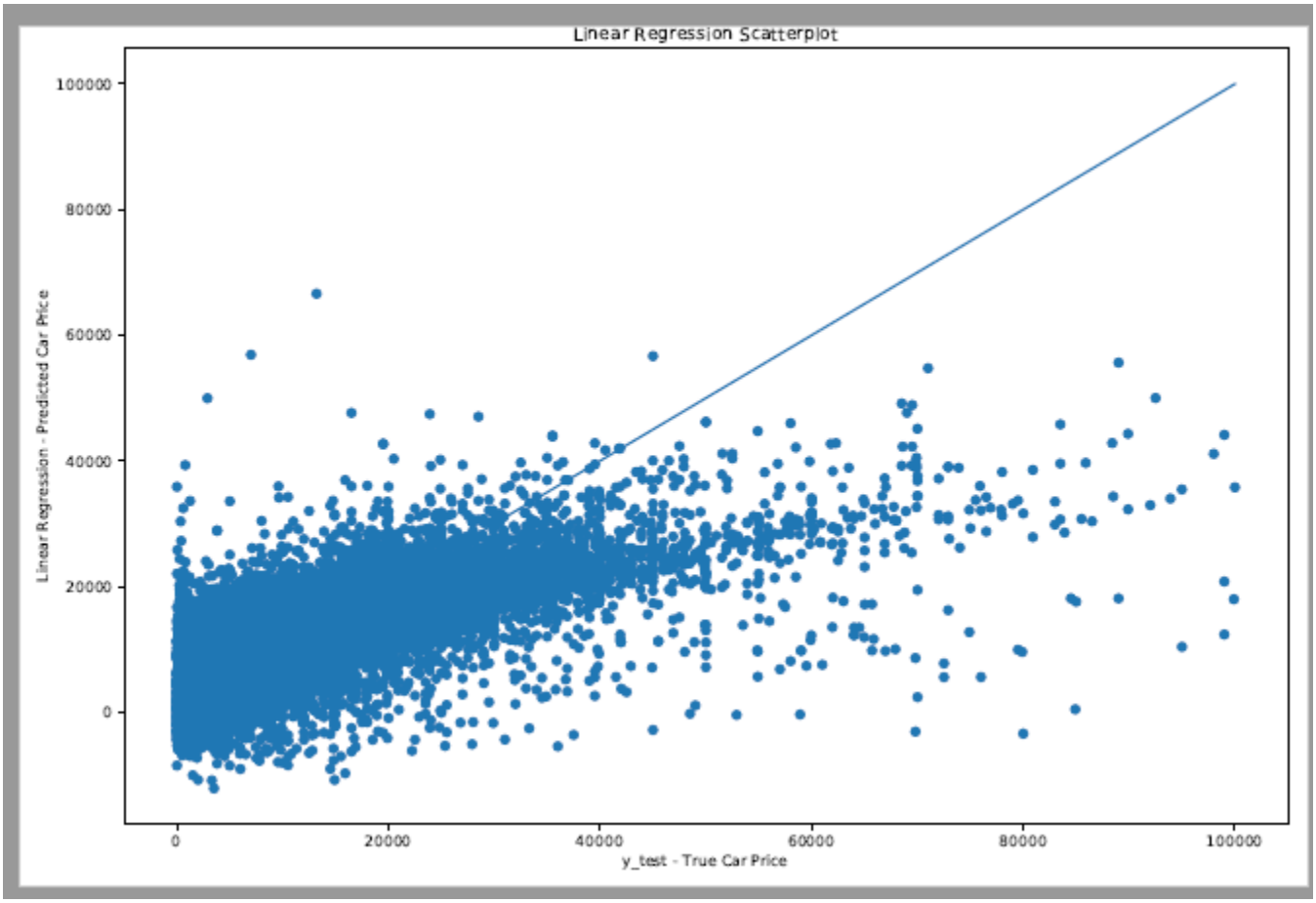

Figure 13: Scatterplot of Multiple Linear Regression model predicted car prices versus true car prices

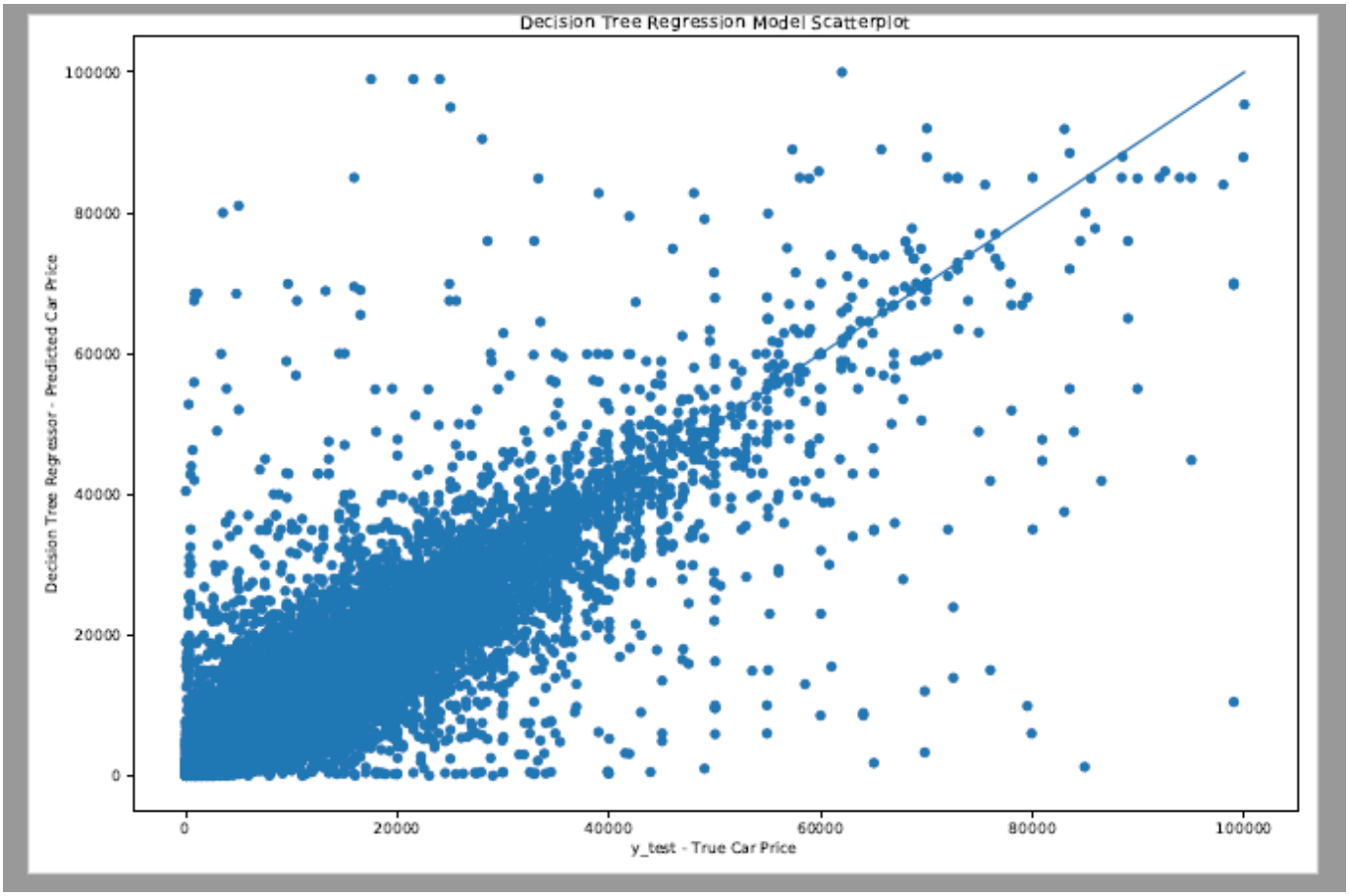

Figure 14: Scatterplot of Decision Tree Regression model predicted car prices versus true car prices

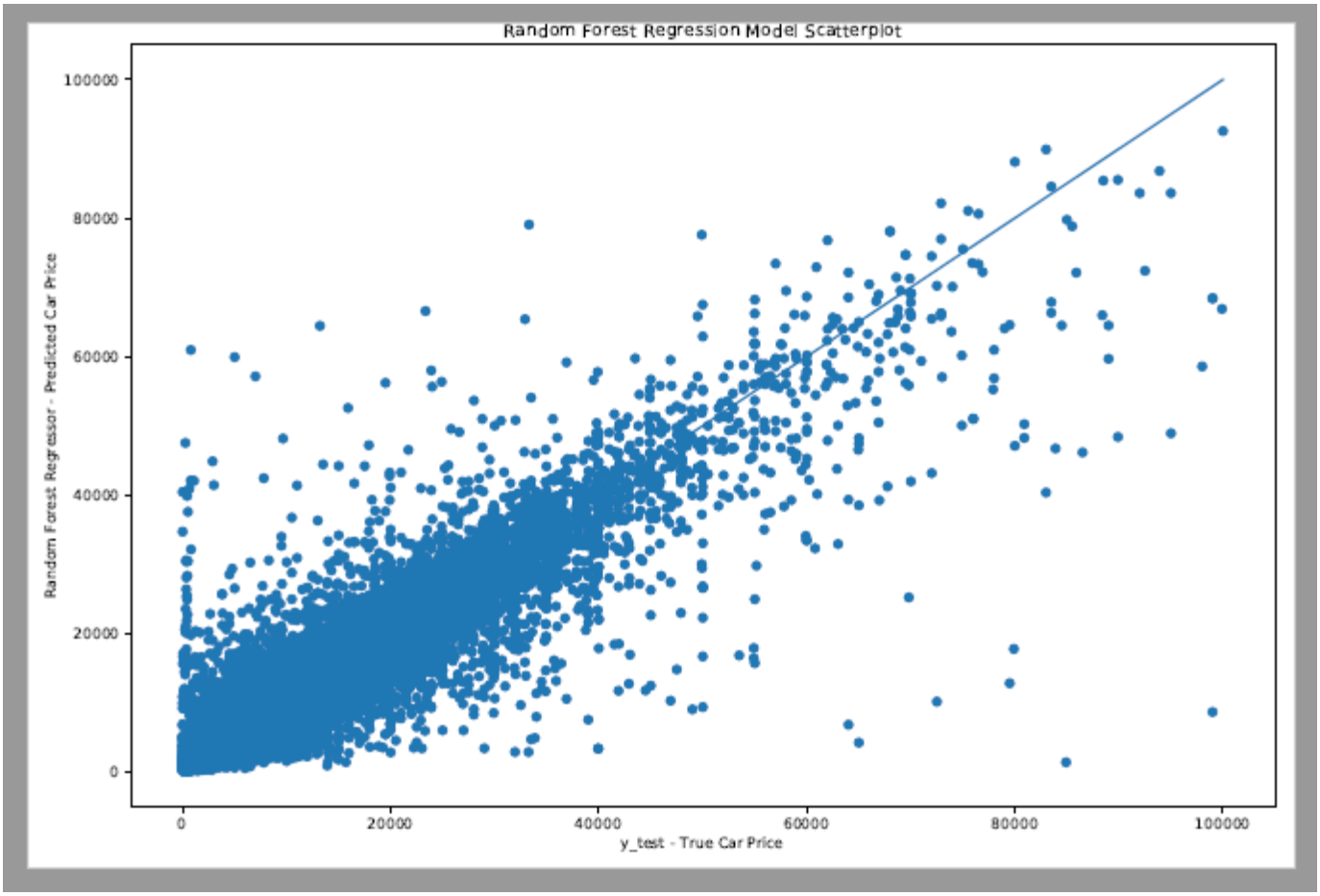

Figure 15: Scatterplot of Random Forest Regression model predicted car prices versus true car prices

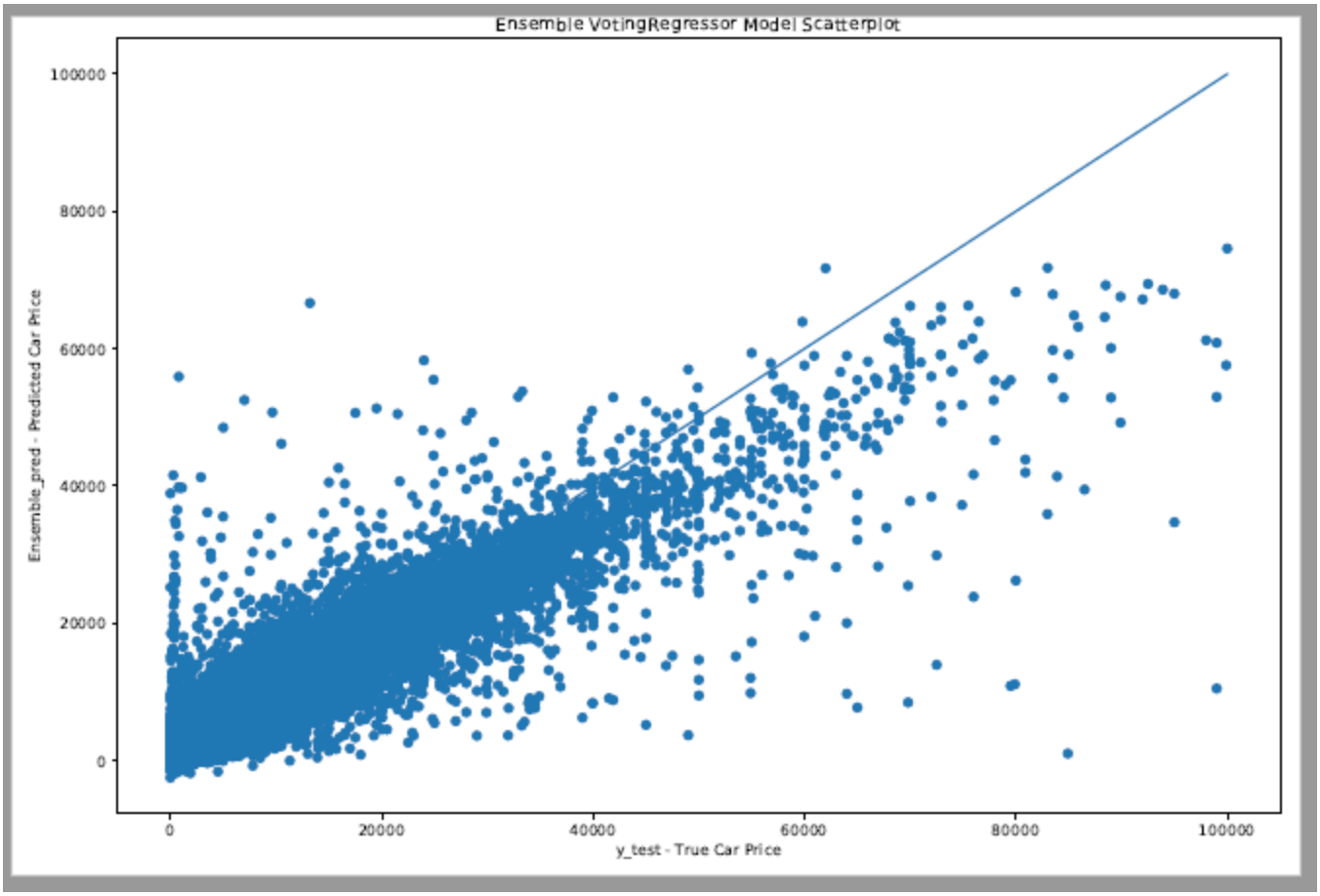

Figure 16: Scatterplot of Ensemble Voting Regressor model predicted car prices versus true car prices

It is worth noting that each scatterplot visually highlights how R-Squared values denote the scatter around the regression line.

Cross-Validation

The train-test split based on one split is known as classical approach. The results above are based on this approach. In this project, the technique of cross-validation is further applied to the splitting of data. This project utilized 10 splits in cross validation. Cross-Validation would reveal consistency in performance of the machine learning regression model and data or reveal inconsistency for further investigation. The followings

are the R-Squared, MAE, MSE, and RMSE model evaluation metrics generated by the four models when than one split is done in cross validation:

R Squared 10-Times Results for Multiple Linear Regression with Cross Validation

```
In [263]: # Display the Scores - R Squared displayed 10 Times
          LR_Scores

Out[263]: array([0.61292635, 0.61918727, 0.62321723, 0.60366085, 0.62024098,
                 0.61992556, 0.61882844, 0.6008811 , 0.62021083, 0.60151036])
```

Figure 17: Multiple Linear Regression Model 10 Times R-Squared Results

R Squared for Multiple Linear Regression with Cross Validation

```
In [339]: # Multiple Linear Regression with Cross Validation R Squared Metric.
          # R Squared has a value between 0 and 1
          # Determine the goodness of fit of predicted used car prices to the true values.
          LR_RSquared_results = cross_val_score(Linear_Regressor, X, y, cv=10, scoring='r2')
          print("R Squared of MLR with Cross Validation is: %0.2f (+/- %0.2f)" % (LR_RSquared_results.mean(), LR_RSquared_results

          R Squared of MLR with Cross Validation is: 0.61 (+/- 0.02)
```

Multiple linear regression with Cross Validation model based on 10 fold splits produced R Squared value of 0.61.

Figure 18: Multiple Linear Regression with Cross-Validation Average R-Squared Result

MAE, MSE and RMSE for Multiple Linear Regression with Cross Validation

```
In [268]: # Cross Validation Linear Regression MAE
          LR_MAE_results = cross_val_score(Linear_Regressor, X, y, cv=10, scoring='neg_mean_absolute_error')
          # Cross Validation Linear Regression MSE and RMSE
          LR_MSE_results = cross_val_score(Linear_Regressor, X, y, cv=10, scoring='neg_mean_squared_error')

          print("MAE of Multiple Linear Regression is: %0.2f (+/- %0.2f)" % (LR_MAE_results.mean(), LR_MAE_results.std() * 2))
          print("MSE of Multiple Linear Regression is: %0.2f (+/- %0.2f)" % (LR_MSE_results.mean(), LR_MSE_results.std() * 2))
          print("RMSE of MLR is: %0.2f (+/-)" % (np.sqrt(-(LR_MSE_results.mean()))), np.sqrt(LR_MSE_results.std() * 2))

          MAE of Multiple Linear Regression is: -3107.03 (+/- 52.32)
          MSE of Multiple Linear Regression is: -25159001.57 (+/- 1674664.20)
          RMSE of MLR is: 5015.87 (+/-) 1294.088174183713
```

The Mean Square Error (MSE) returned by Scikit Learn Cross Validation cross_val_score is always a negative. Hence, the absolute value should be taken or make the negative sign positive before taking the square root to calculate RMSE. Also, the negative sign before MAE and MSE results should be ignored when using these metrics for presentation or description.

Multiple Linear Regression model with cross validation produced MAE metric value of 3107.03, MSE metric value of 25159001.57 and RMSE metric value of 5015.87.

Figure 19: Multiple Linear Regression with Cross Validation Average MAE, MSE and RMSE Results

R Squared 10-Times Results for Decision Tree Regression with Cross Validation

```
In [284]: # Display the Decision Tree Scores - R Squared displayed 10 Times
          DT_Scores

Out[284]: array([0.84555137, 0.82333233, 0.84243773, 0.82563577, 0.82750548,
                 0.82199883, 0.86050558, 0.83059229, 0.84220137, 0.83837112])
```

Figure 20: Decision Tree Regression Model 10 Times R-Squared Results

R Squared for Decision Tree Regression with Cross Validation

```
In [344]: # Decision Tree Regression with Cross Validation R Squared Metric.
          # R Squared has a value between 0 and 1
          # Determine the goodness of fit of predicted used car prices to the true values.
          DT_RSquared_results = cross_val_score(Tree_Regressor, X, y, cv=10, scoring='r2')
          print("R Squared of DTR with Cross Validation Model is: %0.2f (+/- %0.2f)" % (DT_RSquared_results.mean(), DT_RSquared_r

          R Squared of DTR with Cross Validation Model is: 0.84 (+/- 0.02)
```

Figure 21: Decision Tree Regression with Cross Validation Average R-Squared Result

MAE, MSE, and RMSE for Decision Tree Regression with Cross Validation

```
In [289]: # Cross Validation Decision Tree Regression MAE
          DT_MAE_results = cross_val_score(Tree_Regressor, X, y, cv=10, scoring='neg_mean_absolute_error')

          # Cross Validation Decision Tree MSE and RMSE
          DT_MSE_results = cross_val_score(Tree_Regressor, X, y, cv=10, scoring='neg_mean_squared_error')

          print("MAE of Decision Tree Regression is: %0.2f (+/- %0.2f)" % (DT_MAE_results.mean(), DT_MAE_results.std() * 2))
          print("MSE of Decision Tree Regression is: %0.2f (+/- %0.2f)" % (DT_MSE_results.mean(), DT_MSE_results.std() * 2))
          print("RMSE of DTR is: %0.2f (+/-)" % (np.sqrt(-(DT_MSE_results.mean()))), np.sqrt(DT_MSE_results.std() * 2))

          MAE of Decision Tree Regression is: -1448.99 (+/- 39.34)
          MSE of Decision Tree Regression is: -10702669.73 (+/- 1601813.69)
          RMSE of DTR is: 3271.49 (+/-) 1265.627783201991
```

Better results were produced for Decision Tree Regression Model with Cross Validation.

The Mean Square Error (MSE) returned by Scikit Learn Cross Validation cros_val_score is always a negative. Hence, the absolute value should be taken or make the negative sign positive before taking the square root to calculate RMSE. Also, the negative sign before MAE and MSE results should be ignored when using these metrics for presentation or description.

Decision tree regression model with cross validation produced MAE metric value of 1448.99, MSE metric value of 10702669.73 and RMSE metric value of 3271.49.

Figure 22: Decision Tree Regression with Cross Validation Average MAE, MSE and RMSE Results

R Squared 10-Times Results for Random Forest Regression with Cross Validation

```
In [301]: # Display the Random Forest Scores - R Squared displayed 10 Times
          RF_Scores

Out[301]: array([0.89695932, 0.89039955, 0.89793221, 0.87976563, 0.89272822,
                 0.89830128, 0.9052453 , 0.89243783, 0.8941638 , 0.89474127])
```

Figure 23: Random Forest Regression Model 10 Times R-Squared Results

R Squared for Random Forest Regression with Cross Validation

```
In [351]: # Random Forest Regression with Cross Validation R Squared Metric.
          # R Squared has a value between 0 and 1
          # Determine the goodness of fit of predicted used car prices to the true values.
          RF_RSquared_results = cross_val_score(Rd_forest_Regressor, X, y, cv=10, scoring='r2')
          print("R Squared of RF Regression with Cross Validation Model is: %0.2f (+/- %0.2f)" % (RF_RSquared_results.mean(), RF_R

R Squared of RF Regression with Cross Validation Model is: 0.89 (+/- 0.01)
```

Figure 24: Random Forest Regression with Cross Validation Average R-Squared Result

MAE, MSE, and RMSE for Random Forest Regression with Cross Validation

```
In [306]: # Cross Validation Random Forest Regression MAE
          RF_MAE_results = cross_val_score(Rd_forest_Regressor, X, y, cv=10, scoring='neg_mean_absolute_error')

          # Cross Validation Random Forest MSE and RMSE
          RF_MSE_results = cross_val_score(Rd_forest_Regressor, X, y, cv=10, scoring='neg_mean_squared_error')

          print("MAE of Random Forest Regression is: %0.2f (+/- %0.2f)" % (RF_MAE_results.mean(), RF_MAE_results.std() * 2))
          print("MSE of Random Forest Regression is: %0.2f (+/- %0.2f)" % (RF_MSE_results.mean(), RF_MSE_results.std() * 2))
          print("RMSE of RFR is: %0.2f (+/-)" % (np.sqrt(-(RF_MSE_results.mean()))), np.sqrt(RF_MSE_results.std() * 2))

MAE of Random Forest Regression is: -1249.78 (+/- 33.84)
MSE of Random Forest Regression is: -6895795.87 (+/- 1003504.72)
RMSE of RFR is: 2625.98 (+/-) 1001.7508266861528
```

The Mean Square Error (MSE) returned by Scikit Learn Cross Validation cros_val_score is always a negative. Hence, the absolute value should be taken or make the negative sign positive before taking the square root to calculate RMSE. Also, the negative sign before MAE and MSE results should be ignored when using these metrics for presentation or description.

Random Forest Regression model with cross validation produced MAE metric value of 1249.78, MSE metric value of 6895795.87 and RMSE metric value of 2625.98.

Figure 25: Random Forest Regression with Cross Validation Average MAE, MSE and RMSE Results

R Squared 10-Times Results for Ensemble Voting Regressor with Cross Validation

```
In [319]: # Display the Ensemble Voting Regressor Scores - R Squared displayed 10 Times
          VR_Scores

Out[319]: array([0.87058442, 0.86186776, 0.87364356, 0.85044999, 0.86411999,
                 0.87116298, 0.87819753, 0.86215688, 0.86769663, 0.86614426])
```

Figure 26: Ensemble Voting Regressor Model 10 Times R-Squared Results

R Squared for Ensemble Voting Regressor with Cross Validation

```
In [355]: # Ensemble Voting Regressor with Cross Validation R Squared Metric.
          # R Squared has a value between 0 and 1
          # Determine the goodness of fit of predicted used car prices to the true values.
          VR_RSquared_results = cross_val_score(ensemble, X, y, cv=10, scoring='r2')
          print("R Squared of Voting Regressor with Cross Validation Model is: %0.2f (+/- %0.2f)" % (VR_RSquared_results.mean(),

          R Squared of Voting Regressor with Cross Validation Model is: 0.87 (+/- 0.01)
```

Figure 27: Ensemble Voting Regressor with Cross Validation Average R-Squared Result

MAE, MSE and RMSE for Ensemble Voting Regressor with Cross Validation

```
In [324]: # Cross Validation Ensemble Voting Regressor MAE
          VR_MAE_results = cross_val_score(ensemble, X, y, cv=10, scoring='neg_mean_absolute_error')

          # Cross Validation Ensemble Voting Regressor MSE and RMSE
          VR_MSE_results = cross_val_score(ensemble, X, y, cv=10, scoring='neg_mean_squared_error')

          print("MAE of Ensemble Voting Regressor is: %0.2f (+/- %0.2f)" % (VR_MAE_results.mean(), VR_MAE_results.std() * 2))
          print("MSE of Ensemble Voting Regressor is: %0.2f (+/- %0.2f)" % (VR_MSE_results.mean(), VR_MSE_results.std() * 2))
          print("RMSE of EVR is: %0.2f (+/-)" % (np.sqrt(-(VR_MSE_results.mean())), np.sqrt(VR_MSE_results.std() * 2))

          MAE of Ensemble Voting Regressor is: -1592.93 (+/- 40.90)
          MSE of Ensemble Voting Regressor is: -8699389.24 (+/- 1179254.80)
          RMSE of EVR is: 2949.47 (+/-) 1085.9349875617643
```

The Mean Square Error (MSE) returned by Scikit Learn Cross Validation cros_val_score is always a negative. Hence, the absolute value should be taken or make the negative sign positive before taking the square root to calculate RMSE. Also, the negative sign before MAE and MSE results should be ignored when using these metrics for presentation or description.

Ensemble Voting Regressor model with cross validation produced MAE metric value of 1592.93, MSE metric value of 8699389.27 and RMSE metric value of 2949.47.

Figure 28: Ensemble Voting Regressor with Cross Validation Average MAE, MSE and RMSE Results

Table 4: Regression Models with Cross Validation Results Comparison

Evaluation Metrics	Multiple Linear Regression Model	Decision Tree Regression Model	Random Forest Regression Model	Ensemble Voting Regressor Model
R-Squared	0.61	0.84	0.89	0.86
Mean Absolute Error (MAE)	3107.03	1448.99	1249.78	1592.93
Mean Square Error (MSE)	25159001.57	10702669.73	6895795.87	8699389.24
Root Mean Square Error (RMSE)	5015.87	3271.49	2625.98	2949.47

From the results in Table 4 above, Random Forest Regression with the incorporation of cross validation produced the best metric results. It yielded an average R Squared result of 0.89, Mean Absolute Error (MAE) of 1249.78, Mean Square Error (MSE) of 6895795.87 and Root Mean Square Error of 2625.98. Hence, it is the regression model selected for predicting the price of used cars.

With the incorporation of cross validation into construction the four models, the followings are the Scatterplots depicting the four models predicted car prices versus the true car prices:

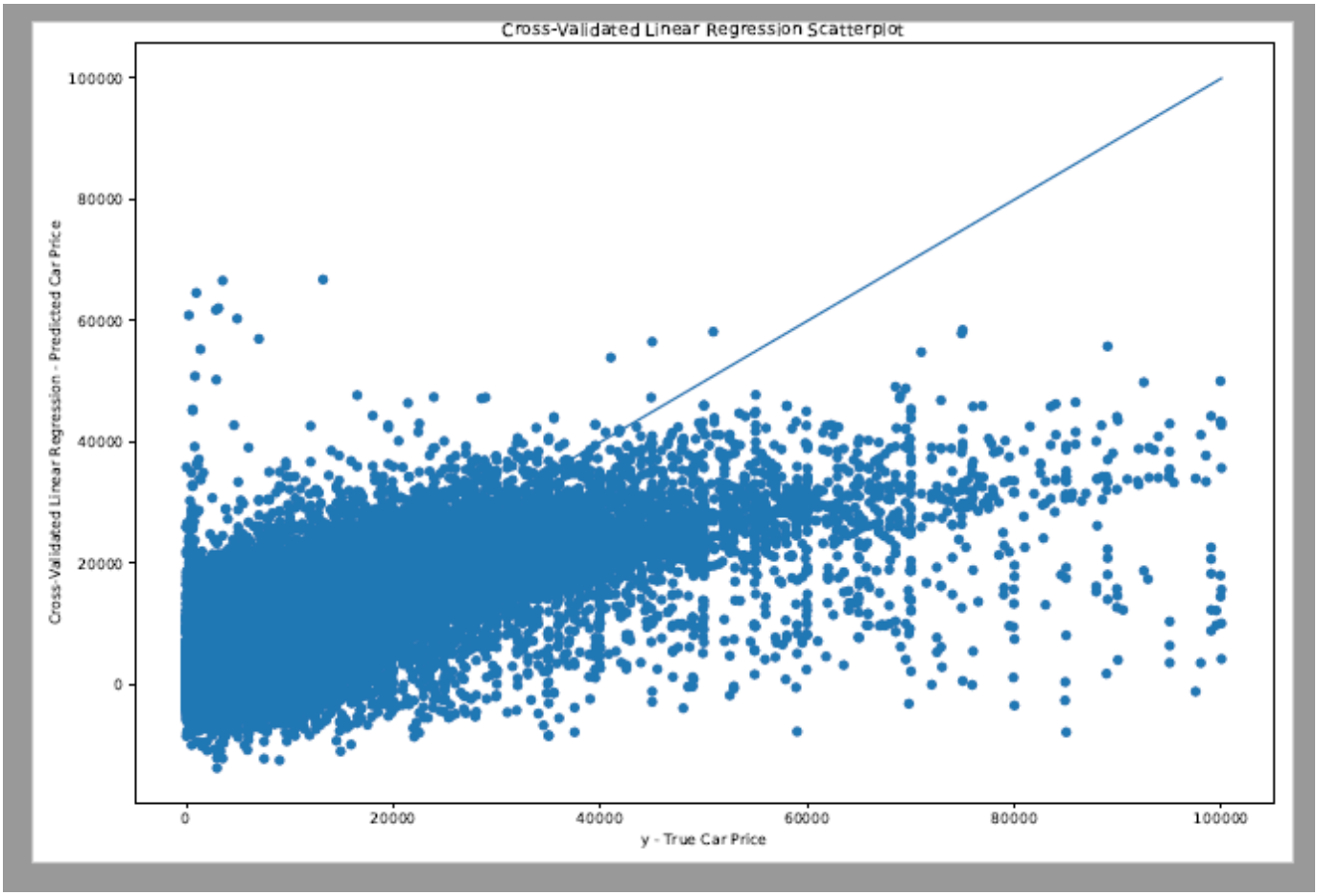

Figure 29: Scatterplot of Multiple Linear Regression with cross validation model predicted car prices versus true car prices

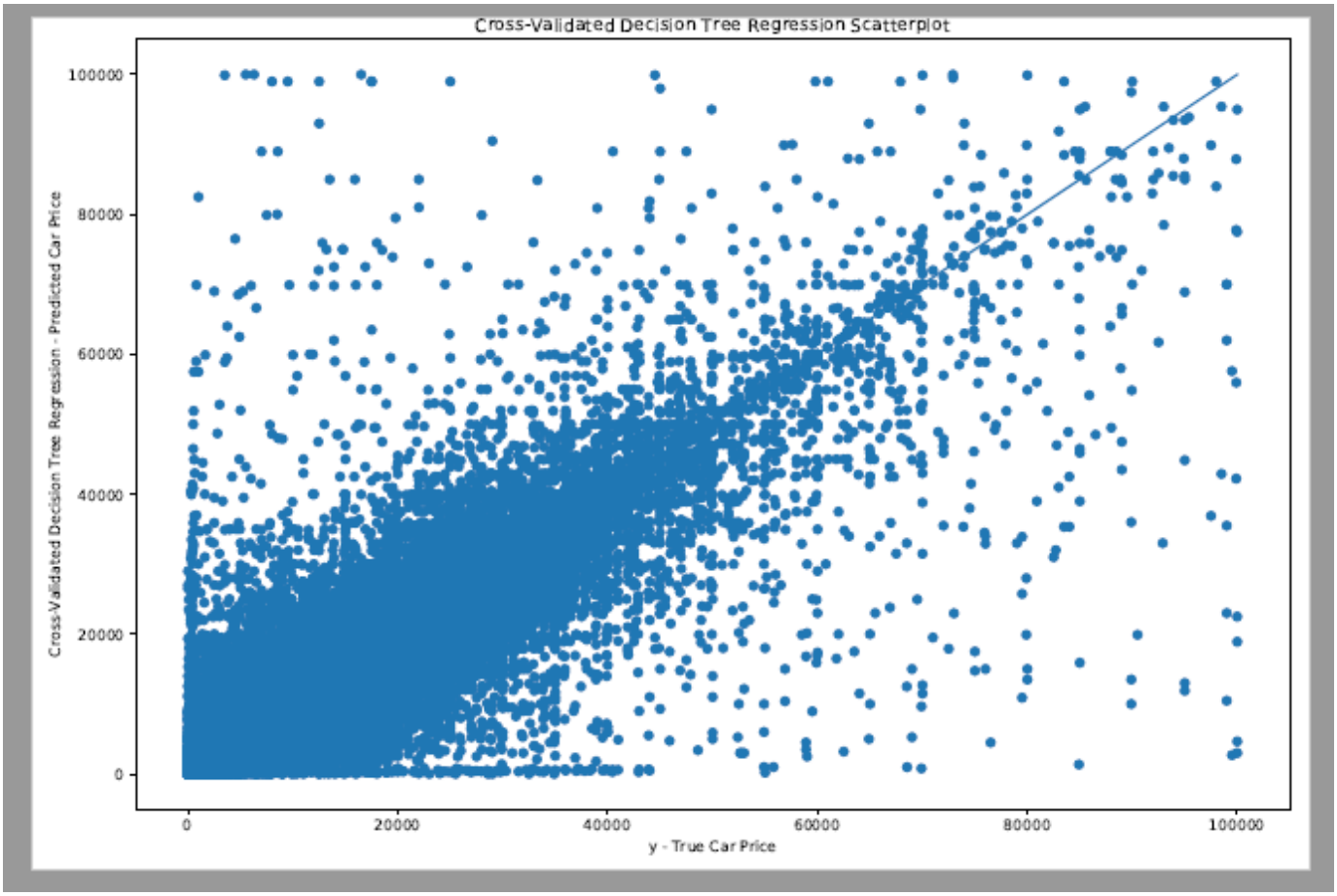

Figure 30: Scatterplot of Decision Tree Regression with cross validation model predicted car prices versus true car prices

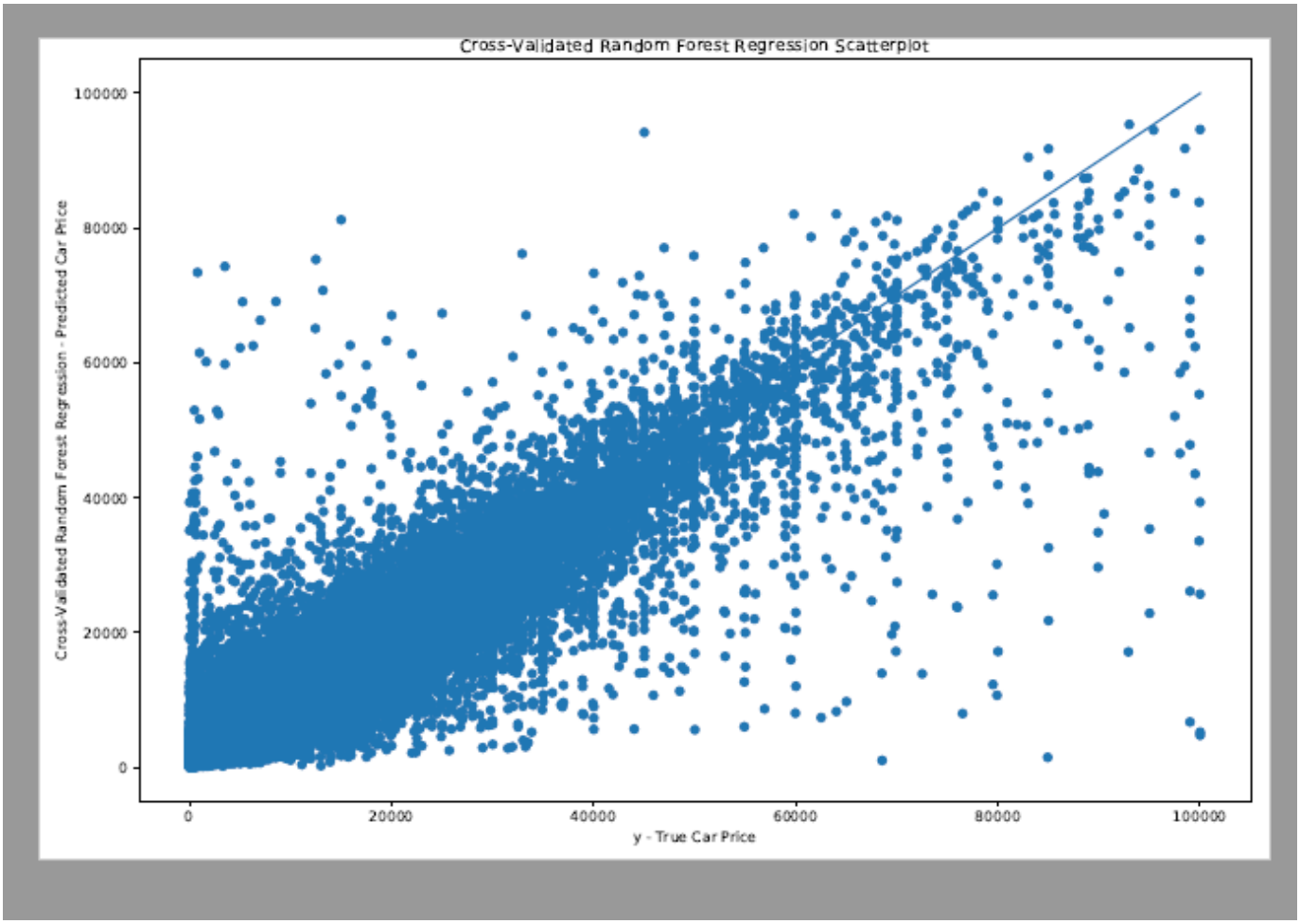

Figure 31: Scatterplot of Random Forest Regression with cross validation model predicted car prices versus true car prices

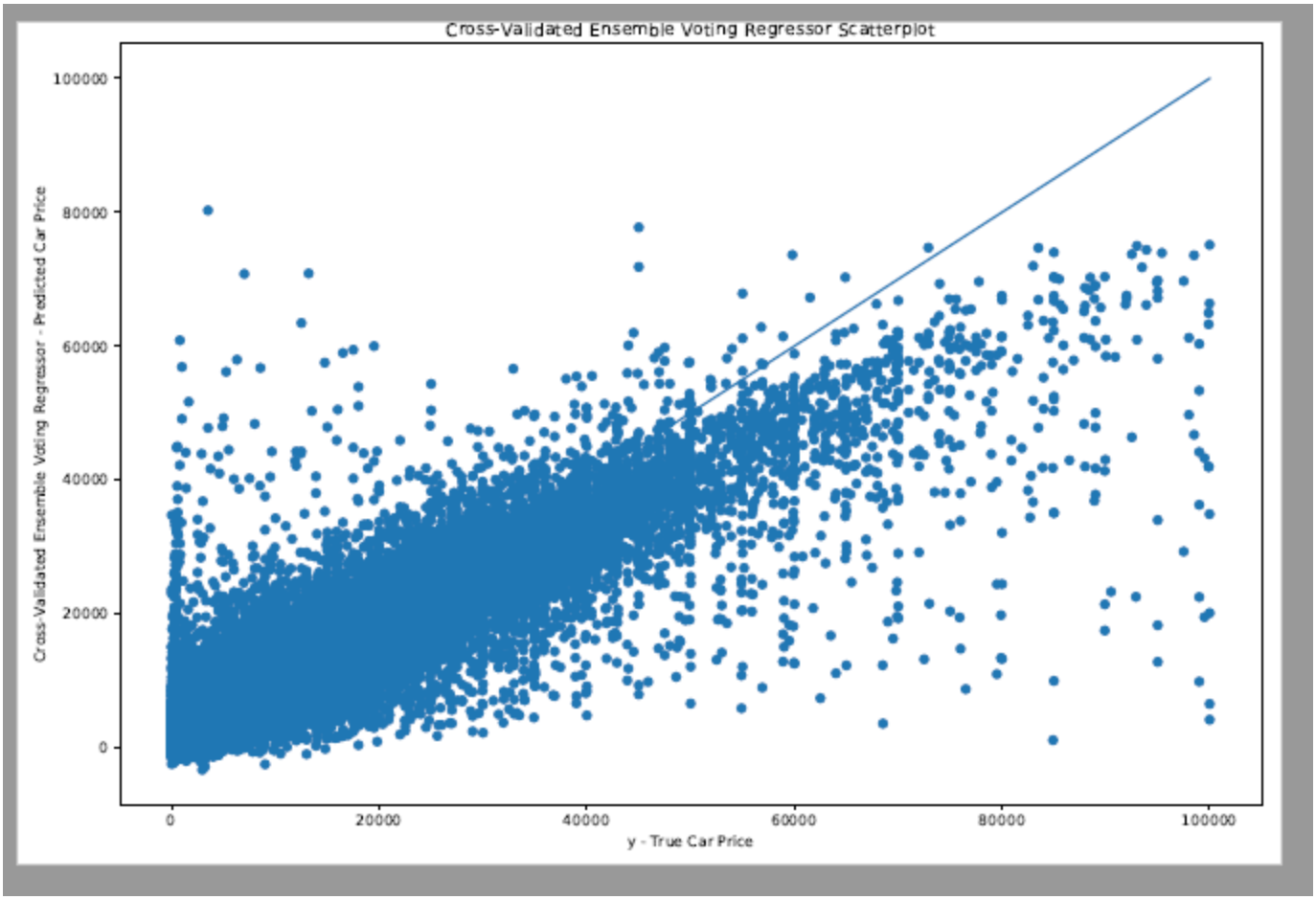

Figure 32: Scatterplot of Ensemble Voting Regressor with cross validation model predicted car prices versus true car prices

6 Conclusion

Multiple Linear Regression (MLR), Decision Tree Regression (DTR), Random Forest Regression (RFR) and Ensemble Voting Regressor are four popular strategies for machine learning and regression. In this project, these four machine learning techniques were leveraged to develop four models for predicting the price of used cars based on Kaggle dataset located at https://www.kaggle.com/orgesleka/used-cars-database. Four evaluation metrics including R Squared, Mean Absolute Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE) were used to select the best model for predicting the price of used cars.

After performing the data cleaning technique on the data, both missing values and outliers were removed from the dataset. Then the data was split into 70/30 train to test ratio. This was utilized in constructing the four models using the machine learning regression techniques. From the results obtained from these four models, Random Forest Regression model produced the best results based on the four-evaluation metrics. Random Forest Regression model generated R-Squared metric of 0.89, Mean Absolute Error (MAE) of 1288.38, Mean Square Error (MSE) of 7154066.55 and Root Mean Square Error metric of 2674.71.

To further showcase the consistency about the performance of the four machine learning regression algorithms used in this project, cross validation technique was incorporated to the data splitting process. Ten splits in cross validation was utilized in this project. The four machine learning regression techniques were run with the features data and target data ten times and the results produced were measured each time. Four models leveraging cross validation were created using these four regression techinques. Again, Random Forest Regression with the incorporation of cross validation produced the best metric results. It yielded an average R Squared result of 0.89, Mean Absolute Error (MAE) of 1249.78, Mean Square Error (MSE) of 6895795.87 and Root Mean Square Error of 2625.98. Hence, it is the regression model selected for predicting the price of used cars.

This project further showcases Car Power Engine, Car Mileage, and the Age of the car as the three most important features in predicting the price of used cars based on the results obtained from feature selection.

7 Future Work

Future work on this project would include predicting the price of each brand and model of used car collections in this dataset. It would be interesting to learn how the inclusion of pictures into the dataset would influence predicting the price of used cars.

References

Brownlee J. May 2016. Metrics To Evaluate Machine Learning Algorithms in Python. Retrieved from https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/

Cross Validation and Model Selection. Retrieved from https://www.pythonforengineers.com/cross-validation-and-model-selection/.

Cross-validation: evaluating estimator performance. Retrieved from https://scikit-learn.org/stable/modules/cross_validation.html.

Frost J. 2019. How To Interpret R-squared in Regression Analysis. Retrieved from https://statisticsbyjim.com/regression/interpret-r-squared-regression/.

Ng R. 2019. Cross-Validation. Retrieved from https://www.ritchieng.com/machine-learning-cross-validation/.

Shaikh R. October 2018. Feature Selection Techniques in Machine Learning with Python. Retrieved from https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e.

https://stackoverflow.com/questions.

https://www.kaggle.com/orgesleka/used-cars-database/discussion

https://www.kaggle.com/orgesleka/used-cars-database/kernels

https://www.researchgate.net/publication/319306871_Predicting_the_Price_of_Used_Cars_using_Machine_Learning_Techniques

http://www.temjournal.com/content/81/TEMJournalFebruary2019_113_118.pdf