House Price Prediction

Submitted by:

1. Md. Ashraful Hasan(2019-1-60-219)

2. Fahim Mohammad Adud Bhuiyan(2019-1-60-223)

3. Ashraful Reza Tanjil(2019-1-60-221)

4. Shinly Noman (2018-1-60-078)

CSE 366: Artificial Intelligence

Submitted to:

Lecturer
Department of Computer Science and Engineering
East West University

Submission Date:

December 28, 2022

Introduction:

The Project is on applying a few classifications algorithms on a random dataset and seeing the performance of the model that we have built. So, we have used 5 classification algorithms for our dataset. The algorithms are:- Logistic Regression, SVM, Decision Tree, Confusion Matrix and Random Forest. Our goal is to find the accuracy for the expected and predicted value of the data set and then cross validation to choose the best test data which will give the best result.

Discussion about the dataset:

It is a dataset that has been downloaded from the kaggle, and the data set is about house price prediction. We have a total 81 columns and 1460 rows to work with where we had one class- SellPrice. The price is given in lacs and then it is our duty to build a model which will predict any kind of new data that train data set by its tuples using those 4 models.

About the algorithms:

Logistic Regression :

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

Random Forest Classifier: Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model.

Decision Tree:A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

Confusion Matrix: Confusion Matrix is a performance measurement for machine learning classification. It shows the accuracy rate of the result.It is extremely useful for measuring Recall, Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.

Support vector machines (SVMs): SVMs are a set of supervised learning methods used for classification, regression and outlier detection. The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where the number of dimensions is greater than the number of samples. SVM algorithm finds the points closest to the line from both classes. These points are known as support vectors. Then it computes the distance between the line and support vectors.This distance is called the margin. The main goal is to maximize the margin. The hyperplane which has the maximum margin is known as the optimal hyperplane. SVM mainly supports binary classification

natively. For multiclass classification, It separates the data for binary classification and utilizes the same principle by breaking down multi- classification problems into multiple binary classification problems.

The steps of our problem that we have approached:

Step 1: Import the library function that is will be used through the code
Step 2: Load the train dataset, and read the csv file

Step 3 : Count rows, columns and then print the datasets

Step 4: Plot few histograms, pair plots

Step 5: Draw the confusion matrix

Step 6: Data Preprocessing
Step 7: Split the dataset train and test

Step 8: Train and test the data set

Step 9: Logistic Regression model building and predicting the result on test data

Step 10: Print the accuracy

Step 11: Cross validate the result

Step 12: Do the same except train and test for each of the model

Step 13: Keep input system for new dataset to classify.

Step 14: Ends the problem

Analysis of the Problem, input and outputs:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
from sklearn import metrics
from mlxtend.plotting import plot_confusion_matrix
```

These are the library function used, Matplotlib is used for the plot of graph, sklearn gives some of the library like to test and train the data set into better portions, accuracy_score for the accuracy, then import the train data set, and the most usable ones pd and numpy in short of the pandas for multidimensional array and memory management.

```
df.head(10).style.background_gradient(cmap = "viridis")
```

	OverallQual	YearBuilt	YearRemodAdd	TotalBsmtSF	1stFlrSF	GrLivArea	FullBath	TotRmsAbvGrd	GarageCars	GarageArea	SalePrice	MSZoning	Utilities	BldgType	Heating	KitchenQual	SaleCondition	LandSlope
0	7	2003	2003	856	856	1710	2	8	2	548	208500	RL	AllPub	1Fam	GasA	Gd	Normal	Gtl
1	6	1976	1976	1262	1262	1262	2	6	2	460	181500	RL	AllPub	1Fam	GasA	TA	Normal	Gtl
2	7	2001	2002	920	920	1786	2	6	2	608	223500	RL	AllPub	1Fam	GasA	Gd	Normal	Gtl
3	7	1915	1970	756	961	1717	1	7	3	642	140000	RL	AllPub	1Fam	GasA	Gd	Abnorml	Gtl
4	8	2000	2000	1145	1145	2198	2	9	3	836	250000	RL	AllPub	1Fam	GasA	Gd	Normal	Gtl
5	5	1993	1995	796	796	1362	1	5	2	480	143000	RL	AllPub	1Fam	GasA	TA	Normal	Gtl
6	8	2004	2005	1686	1694	1694	2	7	2	636	307000	RL	AllPub	1Fam	GasA	Gd	Normal	Gtl
7	7	1973	1973	1107	1107	2090	2	7	2	484	200000	RL	AllPub	1Fam	GasA	TA	Normal	Gtl
8	7	1931	1950	952	1022	1774	2	8	2	468	129900	RM	AllPub	1Fam	GasA	TA	Abnorml	Gtl
9	5	1939	1950	991	1077	1077	1	5	1	205	118000	RL	AllPub	2fmCon	GasA	TA	Normal	Gtl

In the above we got to see our 10 rows and columns with the target that we will be predicting.

```
[ ]  df.columns

Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

These are the targets in fact the classes where the data will fall in.

```
[ ]  df.shape

(1460, 81)
```

Total number of data in the data set, each of the classes have 1460 data so there will be no bias in the dataset.

```
df.info()
```

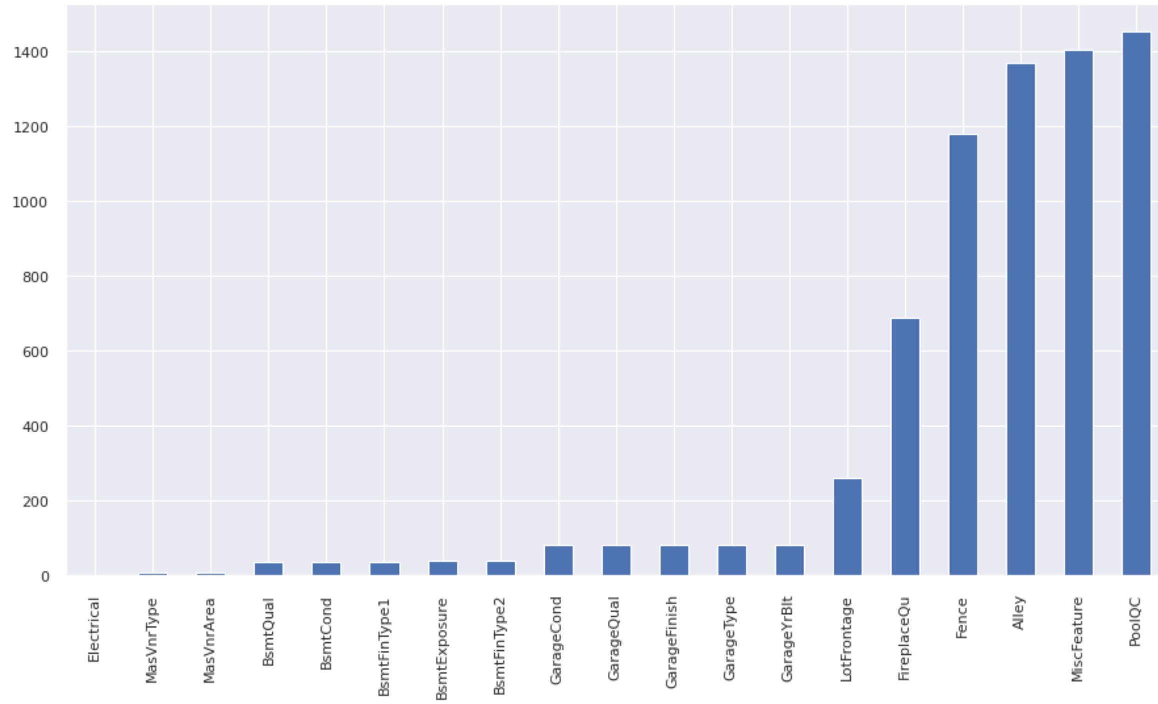
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     1452 non-null   object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
```

```
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

The data types are in float,object and integer.

```
missing=df.isnull().sum()
missing=missing[missing>0]
missing.sort_values(inplace=True)
plt.figure(figsize=(15,8))
missing.plot.bar()
```

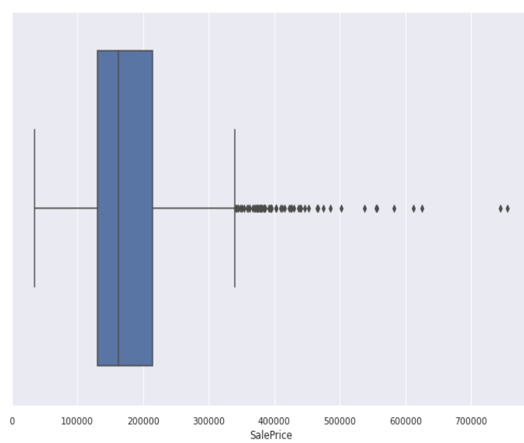
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc15fc074f0>
```



Check whether the dataset has any null value, if there then we have to drop those and count the null value.

```
[ ] df['SalePrice'].describe()
```

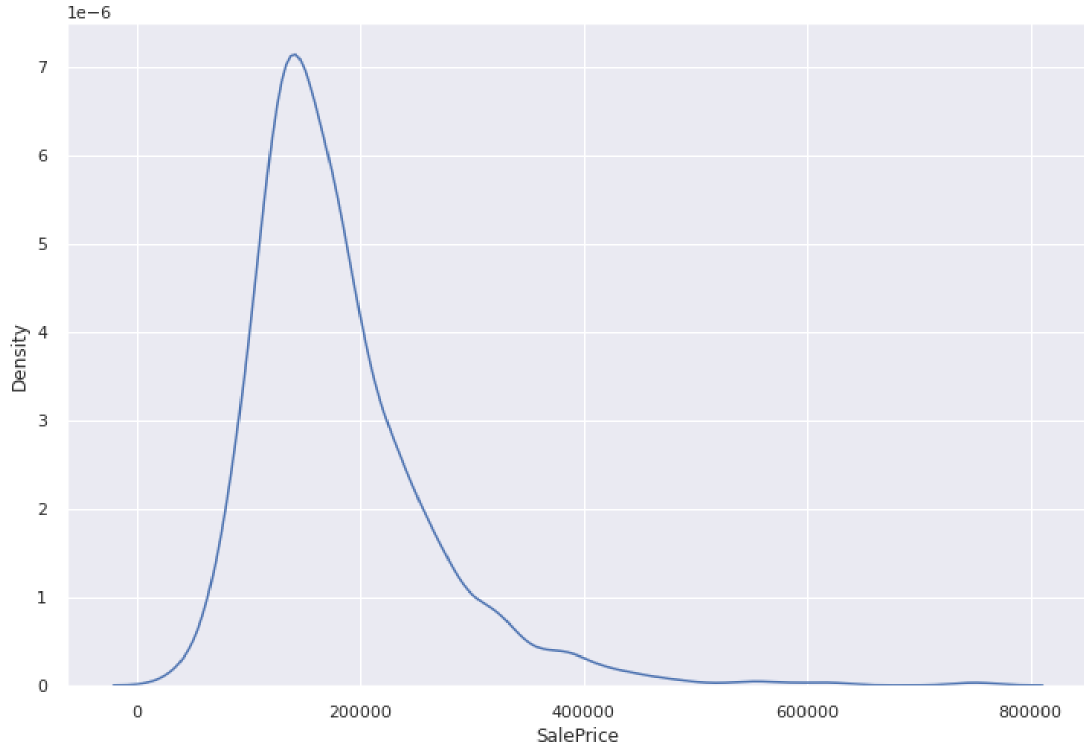
```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```



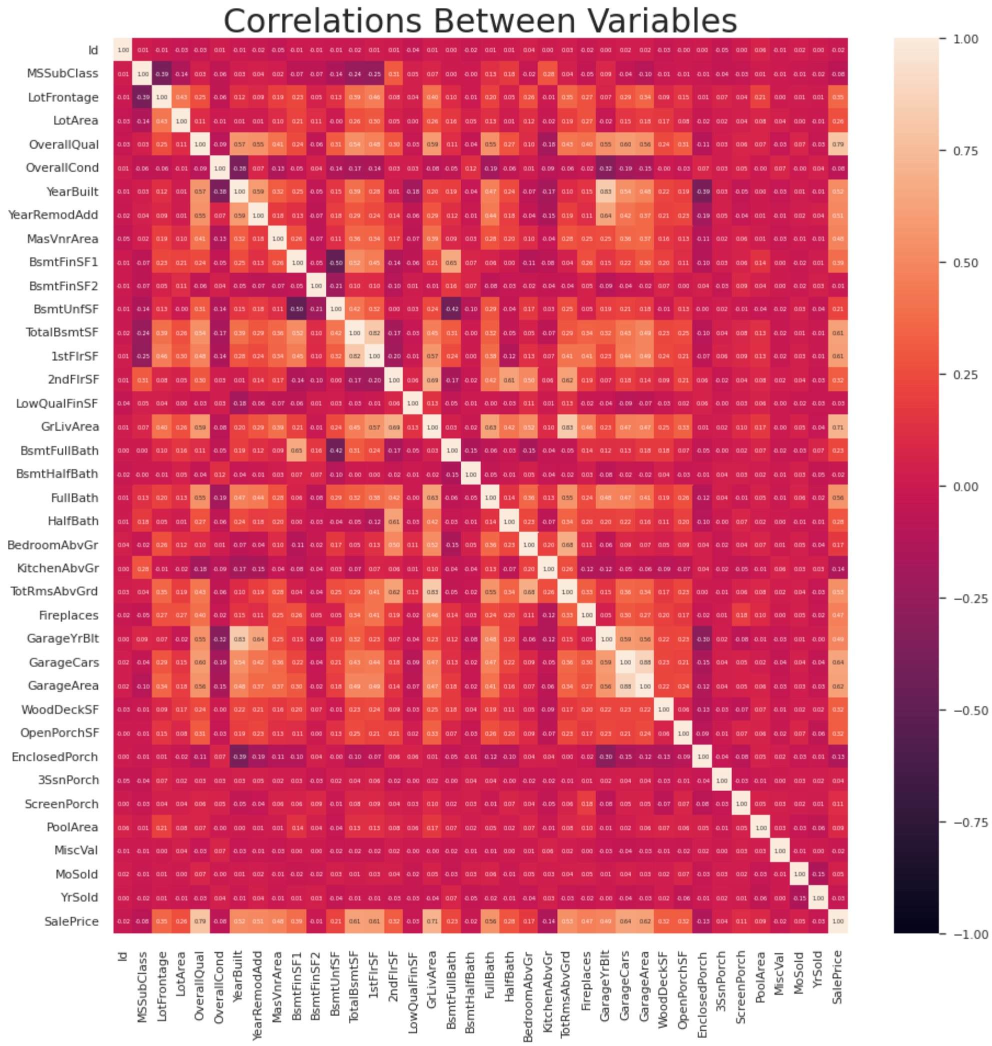
Here is the min, max, standard deviation, mean, median, percentage and Box-Plot for the SalePrice is found.

```
[ ] sns.kdeplot(df['SalePrice'])
```

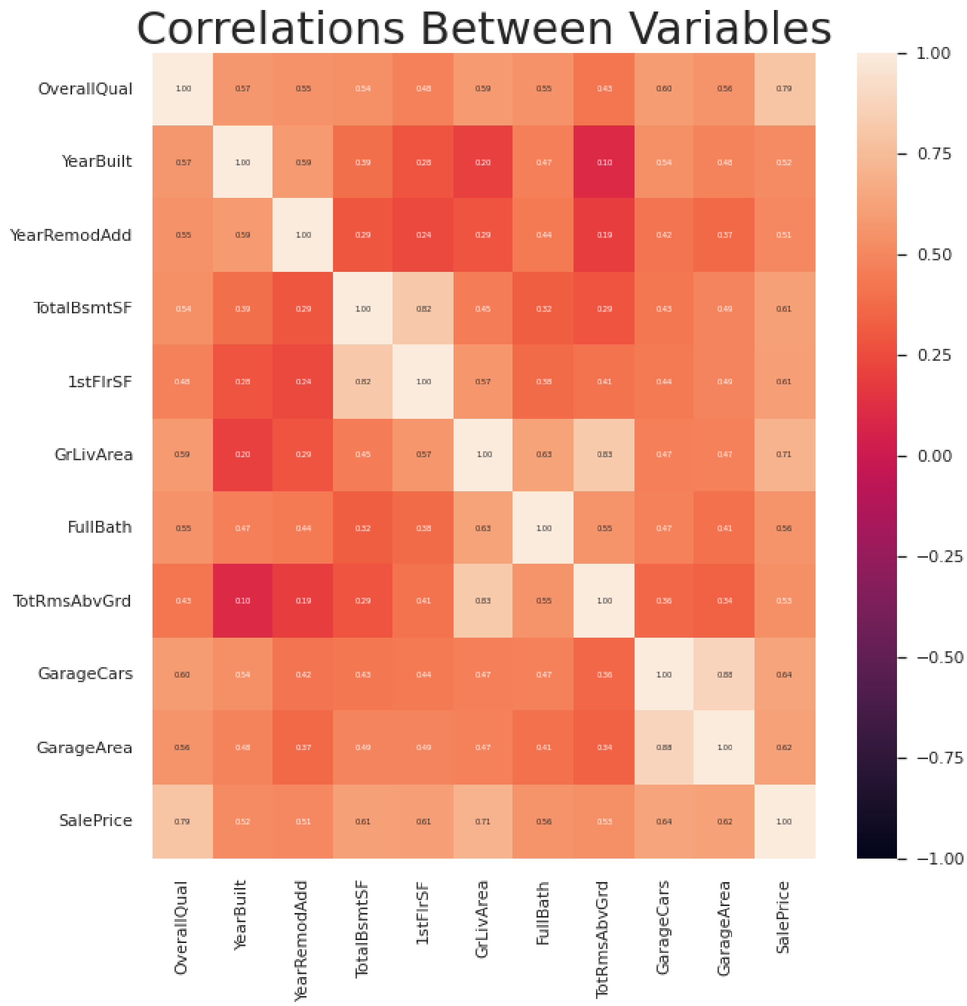
<matplotlib.axes._subplots.AxesSubplot at 0x7fc160d5d9a0>



SalePrice is positively skewed.



Here is the correlation of the full data set.



Visualizing the correlations between numerical variables After Feature Selection and confusion matrix.

```
print(accuracy_score(y_train, y_train_pred))
confusion_matrix(y_train, y_train_pred)
```

```
0.03522504892367906
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=None)
```

```
[ ] print("Shape of train data  ",X_train.shape, y_train.shape)
    print("Shape of test data ", X_test.shape, y_test.shape)
```

```
[ ] # Splitting the data into train and test sets (70:30)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=1, stratify=Y)
```

Before that we have splitted the data into Train and Test, where the train data will have 70 percent and the rest 30 percent is for the test dataset.

Then the 5 models are builded and their results, predictions are:

Regularization: Getting a good accuracy score isn't always pleasant to a data scientist, especially when he gets a good training accuracy along with a poor test accuracy. Though this type of model can predict the training set very well, it can't perform well for the test as well as the new dataset. This type of situation or model is called Overfitting. So how can we solve this problem?

Here comes up Regularization to solve this problem. So in short, we can say that Regularization is the process to prevent a model from being overfitted.

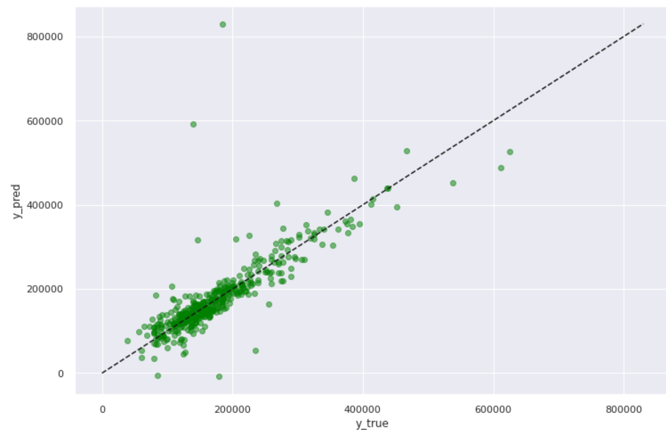
But how? Regularization can be carried out by Ridge Regression, Lasso Regression and Elastic Net Regression. We will try to learn how these regression work and reduce the risk of overfitting throughout this kernel.

Lasso Regression:

1.When you have too many features

2.And you know some of them don't have any significance to your model

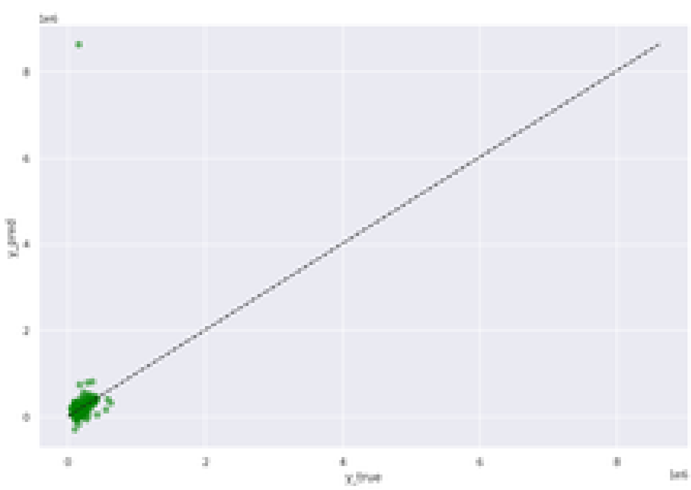
3.When you want to remove the features with less importance



Ridge Regression:

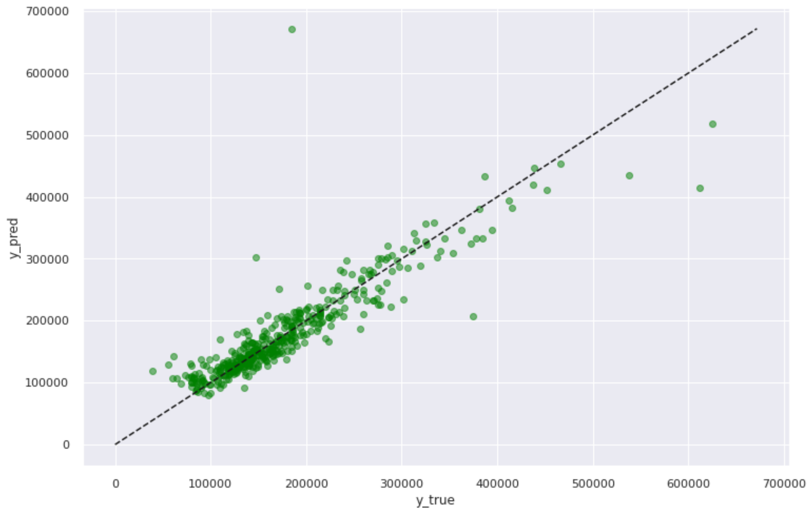
1.When all the features you have are important to your model

2.When you don't want to do feature selection as well as feature removing



Elastic Net Regression:

1. When you don't know whether all the features have significance or not
2. when there are strong correlations between features



SVM:

```
SVM = SVC().fit(X_train, y_train.ravel())
score = SVM.score(X_train, y_train)
print('R_squared:', score)
Y_pred = SVM.predict(X_test)
SVM.score(X_train, y_train)
accuracy_svm = (random_forest.score(X_train, y_train) )
print(accuracy_svm)
```

```
R_squared: 0.08023483365949119
0.9921722113502935
```

Logistic Regression:

```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
acc_log = (logreg.score(X_train, y_train))
acc_log
```

Decision Tree Classifier:

```
[ ] # Decision Tree
    decision_tree = DecisionTreeClassifier()
    decision_tree.fit(X_train, y_train)
    Y_pred = decision_tree.predict(X_test)
    acc_decision_tree = (decision_tree.score(X_train, y_train))
    acc_decision_tree
```

Random Forest Classifier:

```
# Random Forest
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, y_train)
acc_random_forest = round(random_forest.score(X_train, y_train) )
acc_random_forest
```

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (an unknown dataset, for instance from a real problem).

The function used for this cross validation and to find the scores are:

MAE, MSE, RMSE, R2 Score, RMSE (Cross-Validation).

Accuracy score: The fraction of predictions our model got right (number of correct predictions divided by total number of predictions).

Logistic Regression : 0.33072407045009783

Decision Tree : 0.9931506849315068

Confusion Matrix: 0.032

SVM : 0.993

Random Forest: 0.993

Discussion about the drawbacks of the models: The dataset is big and interesting enough to be non-trivial and not slow down experimentation with it.I think a key aspect is that it also teaches about over-fitting. There are not enough columns to give a perfect score: we see this immediately when we look at the scatterplots, and they overlap and run into each other. So any machine-learning approach that gets a perfect score can be regarded as suspicious.

Conclusion: The above data set have 7 features and three classes for observation and then to predict, the 5 models are chosen here for the prediction and its accuracy is same but we could not predict to 100 percent and also there are many other models which may give more better results but through all this exploration of the problem we have gone through a lot of research in websites, documents, articles and many other site which actually helped us to gain the knowledge about major and basic fact of machine learning models , classification algorithm, feature selection, exploratory data analysis, predictions, splitting , validation of dataset including train and tests data. Then some other statistical forms min, max, mean, median, correlation and other things. We also learned to process the data and evaluate the model on our own. This knowledge that we gained from the project will help us to build a good gesture of knowledge in machine learning being a beginner in data science.