Housing Project

Submitted by:

YASH JAISWAL

ACKNOWLEDGMENT

1.1. Linear Models — scikit-learn 1.2.0 documentation

1.11. Ensemble methods — scikit-learn 1.2.0 documentation

1.10. Decision Trees — scikit-learn 1.2.0 documentation

5. Visualizations — scikit-learn 1.2.0 documentation

1.1. Linear Models — scikit-learn 1.2.0 documentation

INTRODUCTION

- Business Problem Framing

Houses are one of the necessary need of each and every person around the globe and therefore housing real esate

Focusing on changing trends in house sales and purchases predictive modelling market mix modelling

Using machine learning in order to predict the actual values of the prospective and decide whether to invest

- Conceptual Background of the Domain Problem

Python knowledge (coding language) which will be used to solve the complete Defaulter project understanding of accuracy ,skewness and basic mathematics / statistical approaches will help to buld an accurate model for this project

- Review of Literature

The major difference between maket value and market price is that the market values . values can crate demand function values along cannot influence price as supply incrases and demand decreses price goes and values not influential as supply decrses and demand increases that price will rise

- Motivation for the Problem Undertaken

Data Analysis and improving the skill set

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

1) **Logistic Regression**
2) **XGBOOST**
3) **Ada Boost Regressor**
4) **Gradient Boosting Regressor**
5) **Random Forest**

- Data Sources and the

```
In [27]: df.describe()
```

Out[27]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeck
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	...	1168.000
mean	724.136130	56.767979	70.988470	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	...	96.206
std	416.159877	41.940650	22.437056	8957.442311	1.390153	1.124343	30.145255	20.785185	182.047152	462.664785	...	126.158
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.000
25%	360.500000	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.000
50%	714.500000	50.000000	70.988470	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.000
75%	1079.500000	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.000
max	1460.000000	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000

8 rows × 38 columns

ir formats

- Data Preprocessing Done

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
x_scaled
```

```
array([[0.75      , 0.5       , 0.09178744, ..., 0.25      , 1.        ,
        0.8       ],
       [0.75      , 0.82051282, 0.97101449, ..., 0.25      , 1.        ,
        0.8       ],
       [0.75      , 0.78205128, 0.53743961, ..., 0.25      , 1.        ,
        0.8       ],
       ...,
       [0.75      , 0.        , 0.01328502, ..., 0.75      , 1.        ,
        0.8       ],
       [0.        , 0.23076923, 0.33333333, ..., 0.5       , 1.        ,
        0.8       ],
       [0.75      , 0.5       , 0.25966184, ..., 0.        , 1.        ,
        0.8       ]])
```

- Hardware and Software Requirements and Tools Used

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We got Random Forest also with the best result and after performaning Hyper tuning we finalized the model

- Testing of Identified Approaches (Algorithms)
- **Logistic Regression**
- **XGBOOST**
- **Ada Boost Regressor**
- **Gradient Boosting Regressor**
- **Random Forest**

- Run and Evaluate selected models

```
lr.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#Lets Print Training Score
pred_train=lr.predict(x_train)
print(r2_score(y_train,pred_train))
```

```
0.9018878515748566
```

```
#Lets Print Testing Score
pred_test=lr.predict(x_test)
print(r2_score(y_test,pred_test))
```

```
0.9012920490228242
```

```
xgb.fit(x_train,y_train)
```

```
▾ XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

```
#Lets Print Training Score
pred_train=xgb.predict(x_train)
print(r2_score(y_train,pred_train))
```

```
0.9999610361640991
```

```
#Lets Print Testing Score
train_pred=xgb.predict(x_test)
print(r2_score(y_test,train_pred))
```

```
0.9171206165286743
```

```
ada.fit(x_train,y_train)
```

```
AdaBoostRegressor
AdaBoostRegressor()
```

```
#Lets Print Training Score
pred_train=ada.predict(x_train)
print(r2_score(y_train,pred_train))
```

```
0.8777938928491866
```

```
#Lets Print Testing Score
train_pred=ada.predict(x_test)
print(r2_score(y_test,train_pred))
```

```
0.8567595163831578
```

```
gbr.fit(x_train,y_train)
```

```
GradientBoostingRegressor
GradientBoostingRegressor()
```

```
#Lets Print Training Score
pred_train=gbr.predict(x_train)
print(r2_score(y_train,pred_train))
```

```
0.9661204218689962
```

```
#Lets Print Testing Score
train_pred=gbr.predict(x_test)
print(r2_score(y_test,train_pred))
```

```
0.9297653298310319
```

```
rf.fit(x_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
#Lets Print Training Score
pred_train=rf.predict(x_train)
print(r2_score(y_train,pred_train))
```

```
0.9823826572087677
```

```
#Lets Print Testing Score
train_pred=rf.predict(x_test)
print(r2_score(y_test,train_pred))
```

```
0.9155465853013497
```

- Key Metrics for success in solving problem under consideration

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

```
y_pred=ridge_model.predict(x_test)
```

```
#MAE
mean_absolute_error(y_test,y_pred)
```

```
32.450306050931566
```

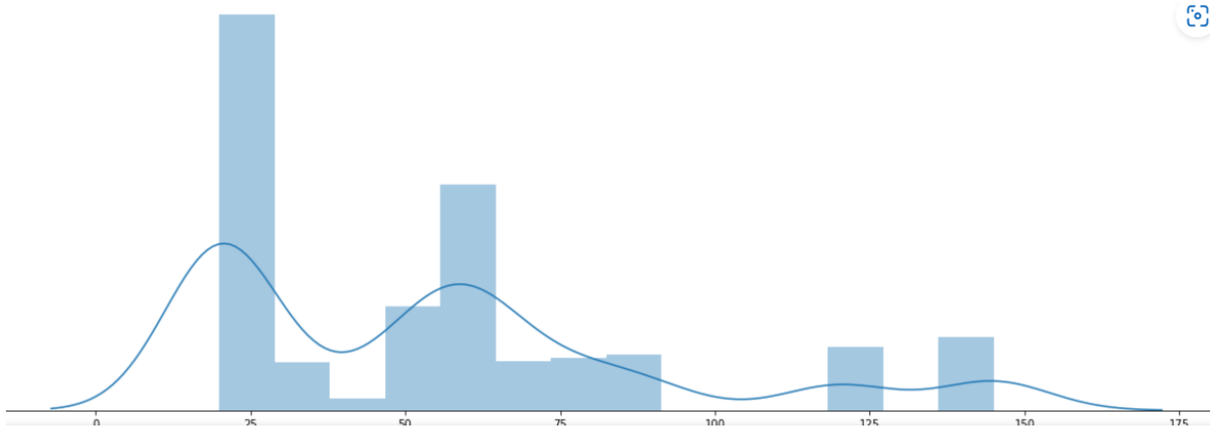
```
#MSE
mean_squared_error(y_test,y_pred)
```

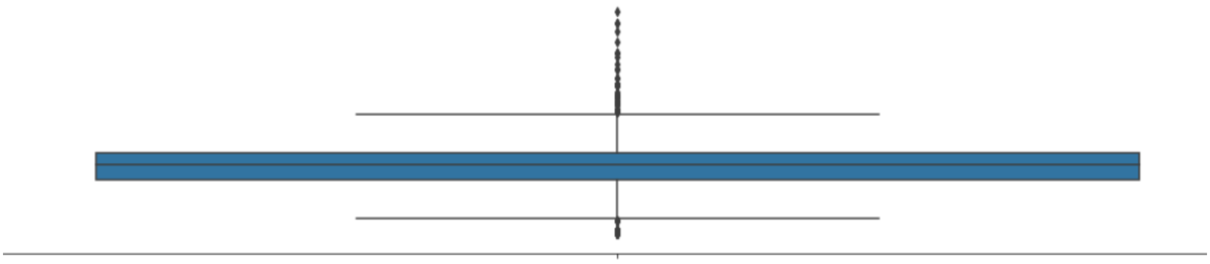
```
1865.1610432956481
```

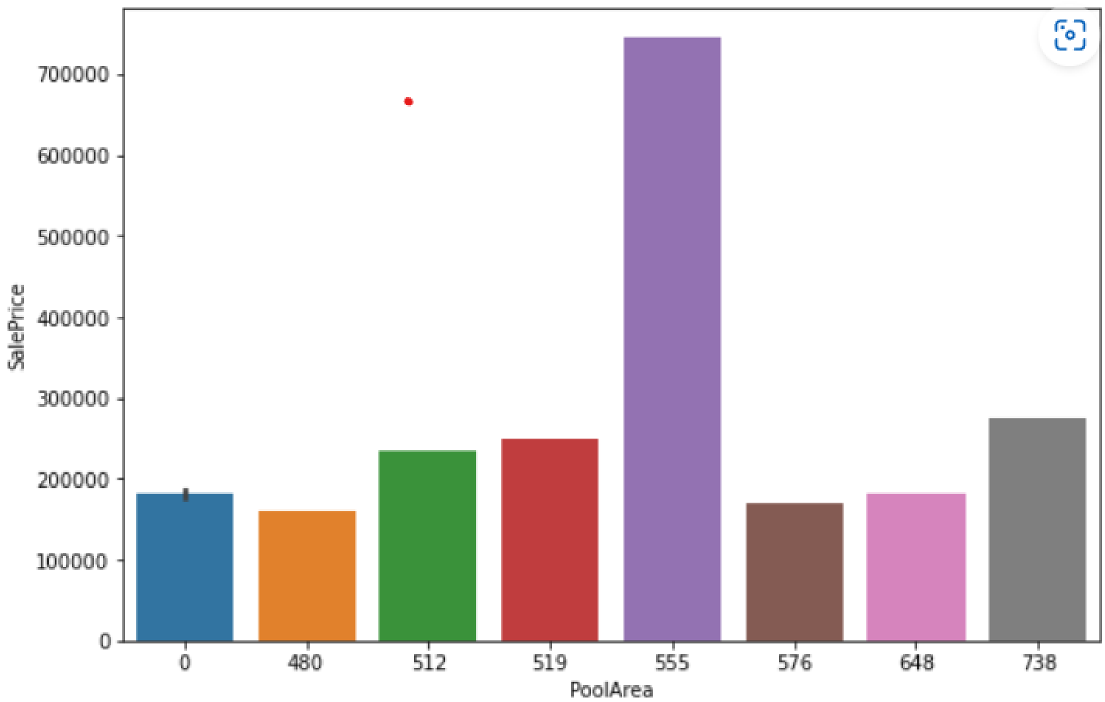
```
#RMSE

np.sqrt(mean_squared_error(y_test,y_pred))
```

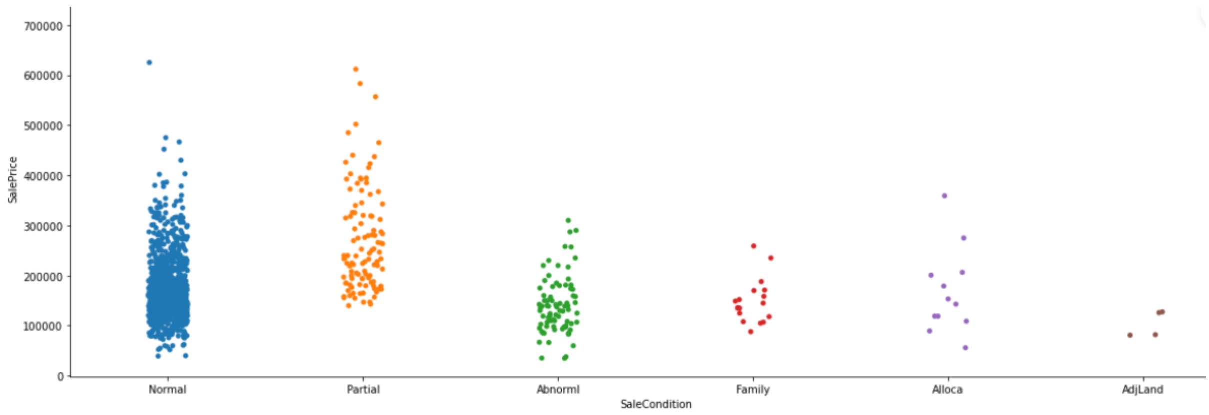
```
43.18751026970237
```

- Visualizations


0
25
50
75
100
125
150
175




SalePrice
700000
600000
500000
400000
300000
200000
100000
0
0
480
512
519
555
576
648
738
PoolArea



CONCLUSION

- Key Findings and Conclusions of the Study

Random forest Algorithm it provides 89% Accuracy which is better then other .

- Learning Outcomes of the Study in respect of Data Science

We used different metrics to check which models best fits the prediction for the dataset

- Limitations of this work and Scope for Future Work

Results is dependent on the data

Thank You