HOUSING PROJECT

Submitted by:

Sristi Kushwaha

ACKNOWLEDGMENT

First of all, I would like to thank Flip Robo Technology who choosed me as an intern in their organization.

Secondly, I would like to thank my mentor Sapna Verma mam who is from Flip Robo Technology and guided me all along the project assigned to me.

And a very warm thank to other mentors of Flip Robo Technology who taught me how to interact with the organization and their working process which is very basic and necessary.

INTRODUCTION

- **BUSINESS PROBLEM FRAMIMG:**

  Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

- **CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM:**

  A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The company is looking at prospective properties to buy houses to enter the market. We are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not.

- **MOTIVATION FOR THE PROBLEM UNDERTAKEN:**

  We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

ANALYTICAL PROBLEM FRAMING

- **MATHEMATICAL/ANALYTICAL MODELING OF THE PROBLEM:**

  We start with doing exploratory data analysis. First, we got some data insights from it. We came to know about different types of columns. We drop some of the columns which are not required. After that we did visualizations to analyze about the effect of features on our target variable. We removed the outliers and skewness of data so that accuracy would not get affected by these problems. After above procedures, we splitted the data into training part and testing part and apply different-2 models so and for evaluation we used some evaluation matrices.

- **DATA SOURCE AND THEIR FORMATS:**

  Dataset is provided by our organization it is in csv format. We have two dataset one is for training and other is for testing. Train dataset contains 1168 rows and 81 columns and test dataset contains 298 rows and 80 columns.

- **DATA PROCESSING DONE:**

  Since data is not so clean for that we have to follow some step to clean it. First of all, checking for the null values and its datatype. This dataset contains many null values in different-2 columns. So, we handle null values by imputing its respective values and some columns contain very large number of null values so we dropped those columns. After that we checked for outliers and skewness of data here both were present so we use IQR method to remove the outlier and power transform to remove the skewness. After data cleaning we go for splitting of data for training and testing.

- **H/W AND S/W REQUIREMENT AND TOOLS USED:**

  We used PCs as hardware tool. For software we used Anaconda prompt in which we used jupyter notebook. In jupyter notebook we used many

libraries like Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn etc. Pandas is used for data structure and their computations, NumPy is used for arrays and some statistical computations, Scikit-learn is used for machine - learning, encoding the data, scaling the data etc. Matplotlib is used for plots and graphs, Seaborn is used for different types of plots advanced version of Matplotlib.

MODEL DEVELOPMENT AND EVALUATIONS

- **APPROACH FOLLOWED:**

  I followed few steps to solve this problem such as knowing about the data and its features and their stats like mean, median, standard deviation and some distribution plot so that I can identify their distribution along the axis. Next is finding if there are any outliers present or not as well as skewness so that we can remove those problems. We split the data into training part and testing part. We scale the data with MinMaxScaler so that data would be ready for model training. And then train the model and compare their accuracy with the help of r2 score and root mean squared error as this was a regression problem. On the basis of that we finalize our best model.

- **TESTING OF IDENTIFIED APPROACH:**

  There are many algorithms which are used for training and testing the data which are listed below.

  - Linear Regression
  - Decision Tree Regressor
  - Adaboost Regressor
  - KNeighbors Regressor
  - Random Forest Regressor
  - XGBRegressor

  We also used regularization I.e., Lasso and Ridge.

- **RUN AND EVALUATE SELECTED MODELS:**

  We trained different-2 models, here we provide model training and its r2 score, cross validation score and root mean squared error to evaluate the model performance.

Model Training

```
from sklearn.model_selection import cross_val_score
```

```
lr=LinearRegression()
lss=Lasso(alpha=0.001)
rg=Ridge(alpha=0.001)
```

```
lr.fit(x_train,y_train)
predlr=lr.predict(x_test)
print(r2_score(y_test,predlr))
print(np.sqrt(mean_squared_error(y_test,predlr)))
crs=cross_val_score(lr,x,y,cv=4)
print(crs.mean())
```

```
0.8828126159802931
0.11811187013983226
0.8731711314634625
```

Regularization

```
lss.fit(x_train,y_train)
predls=lss.predict(x_test)
print(r2_score(y_test,predls))
print(np.sqrt(mean_squared_error(y_test,predls)))
crs=cross_val_score(lss,x,y,cv=4)
print(crs.mean())
```

```
0.8892378103021439
0.1148282860935016
0.8748420834449486
```

```
rg.fit(x_train,y_train)
predrg=rg.predict(x_test)
print(r2_score(y_test,predrg))
print(np.sqrt(mean_squared_error(y_test,predrg)))
crs=cross_val_score(rg,x,y,cv=4)
print(crs.mean())
```

```
0.8828316135654046
0.11810229602368205
0.8731758691585239
```

Some Other Models

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from xgboost import XGBRegressor
```

```
dtr=DecisionTreeRegressor()
knr=KNeighborsRegressor()
rfr=RandomForestRegressor()
adr=AdaBoostRegressor()
xgb=XGBRegressor()
```

```
model=[dtr,knr,rfr,adr,xgb]

for i in model:
    print('model:',i)
    i.fit(x_train,y_train)
    pred=i.predict(x_test)
    print(r2_score(y_test,pred))
    print(np.sqrt(mean_squared_error(y_test,pred)))
    crs=cross_val_score(i,x,y,cv=4)
    print(crs.mean())
    print('\n')
```

```
model: DecisionTreeRegressor()
0.5346805699011161
0.2353578618566275
0.623804460278665


model: KNeighborsRegressor()
0.7286949358567933
0.17971406444908689
0.7152028686813829


model: RandomForestRegressor()
0.8491413107916497
0.13401033139533236
0.838326014225103


model: AdaBoostRegressor()
0.7968561011926449
0.15550881353067492
0.7986302848886021


model: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
0.8302621808732534
0.142148547109615
0.8263837585622744
```

We did hyper parameter tuning after training all models using grid searchcv to improve the accuracy of the models.

Hyperparameter Tuning using GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
# linear regression

par={'fit_intercept':['True','False'],'normalize':['True','False']}
gcv=GridSearchCV(lr,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4, estimator=LinearRegression(),
             param_grid={'fit_intercept': ['True', 'False'],
                         'normalize': ['True', 'False']})
```

```
gcv.best_params_
```

```
{'fit_intercept': 'True', 'normalize': 'True'}
```

```
lrg=LinearRegression(fit_intercept=True,normalize=True)
lrg.fit(x_train,y_train)
predlrg=lrg.predict(x_test)
print(r2_score(y_test,predlrg))
print(np.sqrt(mean_squared_error(y_test,predlrg)))
crs=cross_val_score(lrg,x,y,cv=4)
print(crs.mean())
```

```
0.882812615980293
0.11811187013983229
0.8731711314634624
```

```
# Lasso

par={'alpha':[0.001,0.01,0.1,1,2,5,10,11]}
gcv=GridSearchCV(lss,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4, estimator=Lasso(alpha=0.001),
             param_grid={'alpha': [0.001, 0.01, 0.1, 1, 2, 5, 10, 11]})
```

```
gcv.best_params_
```

```
{'alpha': 0.001}
```

```
ls=Lasso(alpha=0.001)
ls.fit(x_train,y_train)
predls=ls.predict(x_test)
print(r2_score(y_test,predls))
print(np.sqrt(mean_squared_error(y_test,predls)))
crs=cross_val_score(ls,x,y,cv=4)
print(crs.mean())
```

```
0.8892378103021439
0.1148282860935016
0.8748420834449486
```

```
# Ridge

par={'alpha':[0.001,0.01,0.1,1,2,5,10,11]}
gcv=GridSearchCV(rg,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4, estimator=Ridge(alpha=0.001),
             param_grid={'alpha': [0.001, 0.01, 0.1, 1, 2, 5, 10, 11]})
```

```
gcv.best_params_
```

```
{'alpha': 1}
```

```
rdg=Ridge(alpha=1)
rdg.fit(x_train,y_train)
predrdg=rdg.predict(x_test)
print(r2_score(y_test,predrdg))
print(np.sqrt(mean_squared_error(y_test,predrdg)))
crs=cross_val_score(rdg,x,y,cv=4)
print(crs.mean())
```

```
0.8900390297508088
0.11441221613079332
0.8736846674160755
```

```
# random forest regressor

par={'n_estimators':(100,150,180,200,250),'max_depth':np.arange(1,15),'max_features':['auto','sqrt','log2']}

gcv=GridSearchCV(rfr,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4, estimator=RandomForestRegressor(),
             param_grid={'max_depth': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': (100, 150, 180, 200, 250)})
```

```
gcv.best_params_
```

```
{'max_depth': 10, 'max_features': 'auto', 'n_estimators': 100}
```

```
rf=RandomForestRegressor(n_estimators=100,max_depth=10,max_features='auto')
rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(r2_score(y_test,predrf))
print(np.sqrt(mean_squared_error(y_test,predrf)))
crs=cross_val_score(rf,x,y,cv=4)
print(crs.mean())
```

```
0.8538595621311289
0.13189003282808978
0.8390726088833451
```

```
# adaboost regressor

par={'n_estimators':(50,100,120,150,200),'learning_rate':[0.001,0.02,0.01,0.1],'loss':['linear','square','exponential']}

gcv=GridSearchCV(adr,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4, estimator=AdaBoostRegressor(),
             param_grid={'learning_rate': [0.001, 0.02, 0.01, 0.1],
                         'loss': ['linear', 'square', 'exponential'],
                         'n_estimators': (50, 100, 120, 150, 200)})
```

```
gcv.best_params_
```

```
{'learning_rate': 0.1, 'loss': 'square', 'n_estimators': 200}
```

```
ada=AdaBoostRegressor(n_estimators=200,loss='square',learning_rate=0.1)
ada.fit(x_train,y_train)
predada=ada.predict(x_test)
print(r2_score(y_test,predada))
print(np.sqrt(mean_squared_error(y_test,predada)))
crs=cross_val_score(ada,x,y,cv=4)
print(crs.mean())
```

```
0.778000182335018
0.1625658896602339
0.7884819016111559
```

```
# xgb regressor

par={'n_estimators':[100,150,180,200,250],'max_depth':range(1,6),'eta':[0.01,0.1,0.2,0.3]}
gcv=GridSearchCV(xgb,par,cv=4)
gcv.fit(x_train,y_train)
```

```
GridSearchCV(cv=4,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
                                    colsample_bytree=1, gamma=0, gpu_id=-1,
                                    importance_type='gain',
                                    interaction_constraints='',
                                    learning_rate=0.300000012, max_delta_step=0,
                                    max_depth=6, min_child_weight=1,
                                    missing=nan, monotone_constraints='()',
                                    n_estimators=100, n_jobs=8,
                                    num_parallel_tree=1, random_state=0,
                                    reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=1, subsample=1,
                                    tree_method='exact', validate_parameters=1,
                                    verbosity=None),
             param_grid={'eta': [0.01, 0.1, 0.2, 0.3], 'max_depth': range(1, 6),
                         'n_estimators': [100, 150, 180, 200, 250]})
```

```
gcv.best_params_
```

```
{'eta': 0.01, 'max_depth': 1, 'n_estimators': 250}
```

```
xgr=XGBRegressor(n_estimators=250,max_depth=3,eta=0.1)
xgr.fit(x_train,y_train)
predxgr=xgr.predict(x_test)
print(r2_score(y_test,predxgr))
print(np.sqrt(mean_squared_error(y_test,predxgr)))
crs=cross_val_score(xgr,x,y,cv=4)
print(crs.mean())
```

```
0.8664743028402703
0.12607690625420834
0.8522359849085618
```

From all above training and testing score we concluded our final model which has maximum accuracy and minimum root mean squared error is Linear Regression although Lasso has less RMSE error than Linear Regression but difference of r2 score and cross validation score is bit higher than Linear Regression so to avoid any kind of overfitting we select Linear Regression as our final model.

- **VISUALIZATIONS:**

  We did some different-2 visualizations to know the relationship between independent and dependent variables.

  First, we plot count plot to analyze the how the columns are categorizes within its category. And then distribution plot for numerical columns to analyze the distribution of data along the axis.

  We plot scatter plot between numerical columns and target columns to analyze the how the data are affected the target columns.

  We draw box plot to know about outliers in independent columns, next heatmap to analyze the relationship between independent and dependent variables and of course to know about multicollinearity between independent variables.

- **INTERPRETATION OF THE RESULTS:**

  While visualization of our data we try to know what are the relationships among the data according to that we made assumptions also what will be the further process.

  While preprocessing the data we try to clean the data so that any unwanted noise will not there which affects our prediction accuracy and also, we analyze some statistical properties of data so that we can understand what type of data we have.

  Our final process is modeling where we train our data and test that how much accurate our model can predict. And on the basis of metrices we finalize our best model.

CONCLUSION

- **KEY FINDING AND CONCLUSION OF THE STUDY:**

  To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices. Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analyzing other property types beyond housing development.

- **LEARNING OUTCOMES OF THE STUDY WITH RESPECT OF DATA SCIENCE:**

  While doing this project I learned so much about handling the real time data. During visualization I understand that how data are dependent or independent to each other and what can be the relationship between them. while cleaning the data, I realize one of the most important point that cleaning is very important if we want better model for accurate prediction, and also learned different-2 technique to clean the data. We used many algorithms such as Linear Regression, Decision Tree Regressor, Random Forest Regressor, XGBoost Regressor, Regularization etc. And we finalize the best algorithm as Linear Regressor.

- **LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK:**

  Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analyzed. New analytical techniques of machine learning can be used in property research.