31005 Practical Machine Learning Project

GitHub Repository Link: https://github.com/mkalw/UTS_ML2019_ID12572350_ID12595512

Video Pitch Youtube Link: https://youtu.be/s_7slwuoiNE

Introduction

With a rapidly rising real estate industry, a universal area of interest has always been the factors contributing to house prices. Predicting these prices proves to be a practical problem with high significance as consumers struggle to pinpoint the factors and towns within specific cities that correlate to a highly priced house. Along with the crucial consideration of the neighbourhood that the house is located in - other factors influencing price could include the area of the lot, number of rooms, the quality of the house, facilities included, and more. Thus, using data mining to analyse the influences and their level of correlation of behind house prices is an area of practical significance and has the ability to be implemented and benefit consumers in the real estate market.

Exploration

In acquiring a dataset related to our practical problem of predicting house prices, it was imperative to work with reliable and ethically obtained data. For this reason, the chosen source was Kaggle, owned Google and proven to be an ethical platform for machine learning.

The chosen dataset revolves around attributes of houses sold in Ames, Iowa. Each item is described by specific features inherent to the house along with the Sale Price - our desired prediction output. In order to counteract potential problems in data modelling, the dataset has been analysed and identified to have three data types, which are numeric, categorical and boolean. Through identification of these types of data, we can cater to them later in data modelling. After exploring the dataset, it is evident that the data is of a relatively high quality considering there is no presence of missing values and those cells where data is not applicable has been marked with "0" e.g. the attribute PoolArea has 0's if a pool is not present on the property whilst others include the value of the area of the pool. There are also "NA" values - upon examination, we can conclude these are not missing values but rather indicate the absence of particular house features.

An imperative aspect of data exploration throughout this project was to recognise the significance of each individual variable. This was done in each model that was implemented to obtain sufficient knowledge about their correlation with the Sales Price - visualised summaries are available in the code.

In preparing the data for modelling, there was no apparent need to replace missing values as these cells were already replaced with 0's. The next step in preparing the data for various modelling techniques is to split the data into a training set and testing set - in this case, the 80/20 ratio was chosen, thus 80% of the data was used to thoroughly train the data and 20% was tested on.

Methodology

The practical problem that we are addressing through data mining is the prediction of house prices based on influential factors, essentially this becomes the question - What is the predicted sale price of a house considering specific attributes the house possesses? In order to truly explore the depth to which machine learning can provide accurate output, it has been established that 2 different models will be created to allow for a thorough comparison of the capabilities and pros and cons of each along with the ability to enhance areas of improvement in the other model. In particular, we will be implementing the techniques of linear regression and gradient boosting in Python.

1.Linear Regression

In an initial effort to explore the dataset and identify areas of interest and depth in machine learning, linear regression was used. In the process of applying this model, it came to surface that it was difficult to cater the specific dataset attributes to this model and maybe another model would be more suited. The main concern of implementing a linear regression model to predict house prices is that it assumes linearity between the attributes nad the house prices. This may hold true for the majority of attributes such 'YearBuilt' as generally houses that are built recently are comparatively higher priced than older houses, however this will vary depending on the location of the two houses. Therefore we cannot assume lineraity exists between the attributes and the sale price so a linear regression model would not be the most suitable machine learning technique to implement. We completed this model regardless as it would be beneficial to witness the accuracy of the model and how well it could learn from the data.

As the desired result of this model is to predict a numerical output using the effect of individual attributes, it is fitting to use linear regression, which specialises in forecasting values through the analysis of independent variables and their relationship with the dependent variable. In this case, the dependent variable is Sale Price while all other columns are independent variables. In a simple form, the model is built upon the following equation:

> Y = b0 + b1 * x

where, Y is the target variable, b0 and b1 are the parameters and x is the input.

1. In order to implement Linear Regression in Python, we begin by importing the linear regression model from the scikit learn module.

Importing Libraries and Data Model

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
```

2. The missing values were then analysed to identify attributes that have a high percent of missing values in their columns.

```
missing = data.isnull().sum().sort_values(ascending = False)
missing = missing.reset_index()
missing['Percent'] = missing.iloc[:, 1].apply(lambda x:
x*100/np.sum(missing.iloc[:, 1]))
missing.columns = ['Attributes', 'Missing', 'Percent']
gtz = missing['Missing'] > 0
missing = missing[gtz]
missing
```

3. Variables were chosen dependent upon the missing value percentage and if outliers were present in its scatterplot.

Here we determined the variables we will use based on its effect on the sale price (X variables)

```
year_built = list(data['YearBuilt'])
overall_condition = list(data['OverallCond'])
liv_area = list(data['GrLivArea'])
```

```
lot_area = list(data['LotArea'])
overall_qual = list(data['OverallQual'])
year_remod = list(data['YearRemodAdd'])
wood_deck = list(data['WoodDeckSF'])
total_bsmt = list(data['TotalBsmtSF'])
bsmt_unf = list(data['BsmtUnfSF'])
```

Y variable

```
house_price = list(data['SalePrice'])
```

X variables of test data

```
year_built_test = list(data['YearBuilt'])
overall_condition_test = list(data['OverallCond'])
liv_area_test = list(data['GrLivArea'])
lot_area_test = list(data['LotArea'])
overall_qual_test = list(data['OverallQual'])
year_remod_test = list(data['YearRemodAdd'])
wood_deck_test = list(data['WoodDeckSF'])
total_bsmt_test = list(data['TotalBsmtSF'])
bsmt_unf_test = list(data['BsmtUnfSF'])
print("Variables are: year_built, overall_condition, liv_area, lot_area, overall_qual, year_remod, wood_deck, total_bsmt, bsmt_unf")
```

4. The model was the trained, the predicted values were printed out and the accuracy of the linear regression model was then measured using R2*.

Training the model

```
reg = LinearRegression()
model = reg.fit(array_variables,house_price)
y_pred = model.predict(array_variables_test)
```

Print out the predicted values of test data

```
print("Predicted values: ", y_pred)
```

Measuring the accuracy of the model using R^2

```
model.score(array_variables,house_price)
```

2. Extreme Gradient Boosting

After noticing the limitations and lack of flexibility in the Linear Regression Model and thorough research, we decided to build a more complex Extreme Gradient Boosting Model. Extreme Gradient Boosting (XGBoost) is a frequently used machine learning technique for regression and classification problems as it is known for its accuracy, scalability and fast performance. Submissions using XGBoost have dominated Kaggle competitions which is reflective of its accuracy and reliability. It is also a high performing and scalable model that is equipped to handle large datasets. This model, unlike the Gradient Boosting Model (GBM), controls over-fitting and therefore performs faster.

We decided to boost the Stratified K-Fold algorithm as it involves cross-validation and doesn't partition data in a conventional method. Cross-validation promotes accuracy and results in a better trained model as it utilises the dataset in a better way. Stratified k-fold cross validation refers to separating the dataset proportionately into "folds" that so that the mean response value approximately equates.

1. In order to implement XGBoost and Stratified K-Fold in Python, we begin by importing the both models from the xgboost and the scikit model selection module respectively. The seaborn and matplotlib libraries are also imported to allow for data visualisation as shown below.

Import XGBoost to train the Gradient Boosting model

```
from xgboost import XGBRegressor
```

Import train_test_split() method from sk-learn to split the dataframe into training and validation sets

```
from sklearn.model_selection import train_test_split, RandomizedSearchCV, StratifiedKFold
from scipy.stats import randint, uniform
```

Import Matplotlib for Visualisation of Insights

```
import matplotlib.pyplot as plt
```

Import Seaborn for Data Visualisation

```
import seaborn as sns
sns.set()
```

2. We checked for attributes with a high percentage of missing values and made the executive decision to not consider attributes with missing values of 15% and greater to increase accuracy. This included the attributes PoolQC, MiscFeature, Alley and Fence.

Missing data exploration

```
missing = data.isnull().sum().sort_values(ascending = False)
missing = missing.reset_index()
missing['Percent'] = missing.iloc[:, 1].apply(lambda x:
x*100/np.sum(missing.iloc[:, 1]))
missing.columns = ['Attributes', 'Missing', 'Percent']
gtz = missing['Missing'] > 0
missing = missing[gtz]
```

missing

3. The four attributes identified above were dropped from the dataset. Outliers were identified through scatterplot of each attributes against the sale price and were removed.

Removal of Outlier and Significantly Missing Value Attributes

```
data=data.drop(data[data['1stFlrSF']>4000].index)
ncols = cols.drop(['PoolQC', 'MiscFeature', 'Alley', 'Fence'])
```

4. The process of training the model then begins with setting the parameters and implementing StratifiedKFold.

```
params_var = {
'max_depth': [1, 2, 3, 4, 5],
'gamma': [0, 0.5, 1],
'n_estimators': randint(1, 1001), # uniform discrete random distribution
'learning_rate': uniform(), # gaussian distribution
'subsample': uniform(), # gaussian distribution
'colsample_bytree': uniform() # gaussian distribution }
params_fixed = {
'silent': 1
}
```

The number of splits we implemented is 10.

```
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
```

5. The model is then trained with XGBoost.

Train the model using XGBRegressor

```
model = RandomizedSearchCV(
estimator=XGBRegressor(**params_fixed, seed=seed),
param_distributions=params_var,
n_iter=10,
cv=cv,
scoring='r2',
random_state=seed
)
```

Fit the model over training data

```
model.fit(X, y, eval_set=[(test, actual_y)], verbose=False)
model.best_estimator_
```

6. The sale prices of the test data are then predicted and extracted to compare the actual sale price with the predictions produced by the model.

```
predicted_y = model.predict(test)
print("Actual Price of first 10 instances:\n")
print(actual_y.head(10))
print("\nPredicted Price of first 10 instances:\n")
print(pd.DataFrame(predicted_y).head(10))
```

Evaluation

In implementing both models, there was a need to evaluate the performance of each one and understand what was done well and where the areas of improvement were located. To effectively evaluate and compare both models, an accuracy measure of R Squared will be implemented.

1. Linear Regression

To preserve the quality of the model, we ensured to eliminate variables that had an exceedingly high number of "NA", which in this case was not a missing value but rather a feature which was not present in the house for example, a swimming pool. To minimise disruptions to the model, we chose not to model attributes which exceeded 15% in missing values. This will increase the overall accuracy and relaibility of the model.

In evaluating the model, we can also note an area of disadvantage that we were unable to model categorical features in Linear Regression and only used numerical attributes - this could have potentially been addressed through enhancing the model, however, due to the decision of moving onto Gradient Boosting and the constrained nature of Linear Regression, it did not seem worth the effort. Linear regression prediction models also assume linearity between the attributes and the variable to be predicted. In this case, attributes such as the year the house was built would be considered to have a linear relationship to the sale price therefore an increase in the year built, would produce an equivalent increase in house prices when this may not be the case.

Ultimately, the model provided an accuracy rate of 0.7708. Had we considered and improved the issues discussed above, this measure may have been improved.

2. Extreme Gradient Boosting

Shortly after beginning the Linear Regression model, we realised that Extreme Gradient Boosting would be more fitting to the dataset and aim, hence, we were able to implement a more thorough model through this method. The Extreme Gradient Boosting model does not assume linearity, is scalable and is known for producing highly accurate predictions. This model also addressed the issue of our ability to utilise categorical variables in our machine learning algorithm.

Similar to Linear Regression, the same method was implemented in this model to analyse missing values. Additionally, the removal of an outlier was also conducted to preserve stability of the model.

One aspect of this model is that we implemented a section dedicated to separating categorical and numerical variables - this allowed the model to be trained upon all attributes rather than a select few. Accordingly, this would improve the reliability of the model.

The model provided an accuracy score of 0.8615.

Comparison of Methods

In comparing the 2 models, we can look at the R Squared value. The linear regression model scored 0.77 whilst XGBoost scored 0.86 - in this case, XGBoost proves to be the more accurate model.

As an example, see below the comparison of predicted Sale Price values for each method compared to the actual value.

Actual

> 208500

> 181500

> 223500

XGBoost

> 203311

> 170324

> 208093

Linear Regression

> 144426

> 248979

> 189498

Ethical

The ethical model adopted throughout this project was the Kantian approach which is a deontological theory. The model dictates that actions should be determined by its nature rather than the potential consequences that could eventuate. The decisions made during the project were based on the universal duties and principles including ensuring accuracy through cross-examining two data mining prediction techniques, using all the data attributes in the dataset to ensure accurate and reliable conclusions can be drawn from the results and ensuring that the data was collected ethically and by a reliable source.

We have utilised the techniques, XGBoost Regression (Extreme Gradient Boosting Regression) and Linear Regression to predict house prices. Linear Regression, unlike XGBoost Regression, may produce results that aren't reflective of the nature of the relationship between the data attributes. Linear regression assumes linearity between data elements meaning an increase in an attribute would produce an equivalent increase in house prices therefore the results may be misused by drawing inaccurate conclusions. As assuring accuracy was considered imperative and aligns with our ethical model, we compared the results with the findings produced using the XGBoost Regression. The XGBoost Regression technique is known to be highly reliable, which is reflected in the number of winning Kaggle submissions that utilise XGBoost and is an improvement upon the shortcomings of the GBMs (Gradient Boosting Machines) through an optimised system and enhanced algorithms.

Conclusion

Conclusively, we found it highly beneficial to have developed two very different methods and observe the capabilities of each one - beginning with Linear Regression truly challenged our perspective on the dataset and motivated us to go further into machine learning using XGBoost. An area of improvement that is very transparent is the potential enhancing of the Linear Regression Model to have observed the highest accuracy that could have been achieved and similarly, had there have been more time to explore XGBoost, we could have trained the model. However, we are satisfied with the results and it is evident that there is huge potential in machine learning in the real estate industry.