Computing Science Department Faculty of Science

COMP 4980

MACHINE LEARNING

Fall, 2022

FINAL PROJECT

Student name: Tuan Nguyen

Student ID: T00615922

Project Link:

https://colab.research.google.com/drive/1QiTnQaSroDN7S6dwlpbYxnEe8iTJ1cBU?usp=sharing

Table of Contents

1.DATA DESCRIPTION

1.1 Dataset Information:

Dataset name: Miami Housing 2016

Dataset link: http://bit.ly/3VbwCaA

Publish date: June 16 2022

Author: Leo Grin

Data supplier: Steven C. Bourass.

The dataset can be acquired through OpenML or Kaggle website with the keyword “Miami Housing”.

1.2 Dataset Description

The Data about the housing sale price of 13,932 single-family homes sold in Miami in 2016. The data type is scalar, it has 14 features and 13,932 instances (File size total: 1.63 MB)

There are 14 features including:

PARCELNO: unique identifier for each property. About 1% appear multiple times.

SALE_PRC: sale price ($)

LND_SQFOOT: land area (square feet)

TOT_LVG_AREA: floor area (square feet)

SPEC_FEAT_VAL: value of special features (e.g., swimming pools) ($)

RAIL_DIST: distance to the nearest rail line (an indicator of noise) (feet)

OCEAN_DIST: distance to the ocean (feet)

WATER_DIST: distance to the nearest body of water (feet)

CNTR_DIST: distance to the Miami central business district (feet)

SUBCNTR_DI: distance to the nearest subcenter (feet)

HWY_DIST: distance to the nearest highway (an indicator of noise) (feet)

age: age of the structure

avno60plus: dummy variable for airplane noise exceeding an acceptable level

structure_quality: quality of the structure

month_sold: sale month in 2016 (1 = Jan)

LATITUDE

LONGITUDE

The format of the dataset is ARFF datafile. The dataset doesn't contain any empty values or errors. It was collected after 2016 and published on June 16, 2022. I tried running a few machine learning algorithms, and this data is quite useful because it can help users understand which aspects have the most effect on the price of housing in Miami. In my opinion, the dataset is quite out of date, however, it still can provide to users a more detailed look at the factors that affect home prices in Miami. Due to the fact that, people can remove or add these aspects to their requirement when looking for a house in Miami. Moreover, the analysis of this data will also provide a reference value on the given data.

2. DATA ANALYSIS

The dataset has a total of 14 features including target class. All the data in the dataset is numeric with 2 special data are LATITUDE and LONGITUDE. The table below will show some data statistic:

	LATITUDE	LONGITUDE	LND_SQFOOT	TOT_LVG_AREA	SPEC_FEAT_VAL	RAIL_DIST	OCEAN_DIST
count	13932	13932	13932	13932	13932	13932	13932
mean	25.72881144	-80.3274752	8620.9	2058	9562.5	8348.5	31691.0
std	0.140633285	0.089199072	6070.1	813.5	13891.0	6178	17595.1
min	25.43433337	80.5421721	1248	854	0	10.5	236.1
25%	25.62005604	-80.4032775	5400	1470	810.0	3299.5	18079.4
50%	25.73181003	-80.3389106	7500	1877.5	2765.5	7106.3	28541.8
75%	25.85226854	-80.2580187	9126.3	2471	12352.3	12102.6	44310.7
max	25.97438187	-80.1197463	57064	6287	175020.0	29621.5	75744.9
	Water_Dist	CNTR_DIST	SUBCNTR_DI	HWY_DIST	age	month sold	structure quality
count	13932	13932	13932	13932	13932	13932	13932
mean	11960.3	68490.3	41115	7723.8	30.7	6.7	3.5
std	11933	32008.5	22161.8	6068.9	21.2	3.3	1.1
min	0	3825.6	1462.8	90.2	0	1	1
25%	2675.9	42823.1	23996.3	2998.1	14	4	2

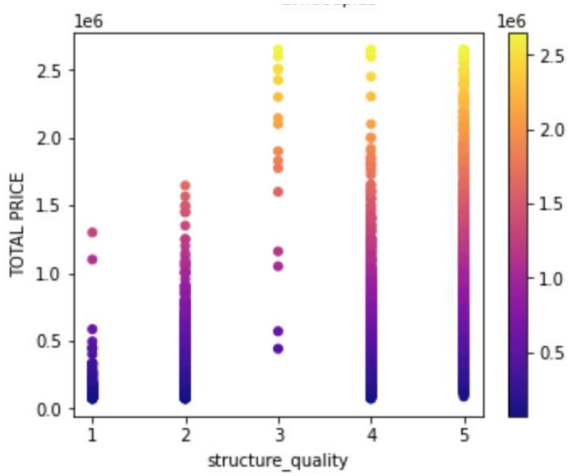
50%	6922.6	65852.4	41109.9	6159.8	26	7	4
75%	19200	89358.3	53949.4	10854.2	46	9	4
max	50399.8	159976.5	110553.8	48167.3	96	12	5

The above table has removed PARCELNO, month_sold, and avno60plus which are not meaningful when we put them in this table. Although the data of LATITUDE and LONGITUDE are not clear and useful in the above table, based on the min and max values we can find the limits of the dataset's area on a world map. Based on the data of LND.SQFOOT (land area: square feet), we can see that 75% of the housing has a land area of less than 9126.3 square feet while the maximum land area of housing is 57064 square feet. The perspectives above are the same with the data of TOT_LVG_AREA (floor area: square feet) and SPEC_FEAT_VAL (value of special features: swimming pool… $).

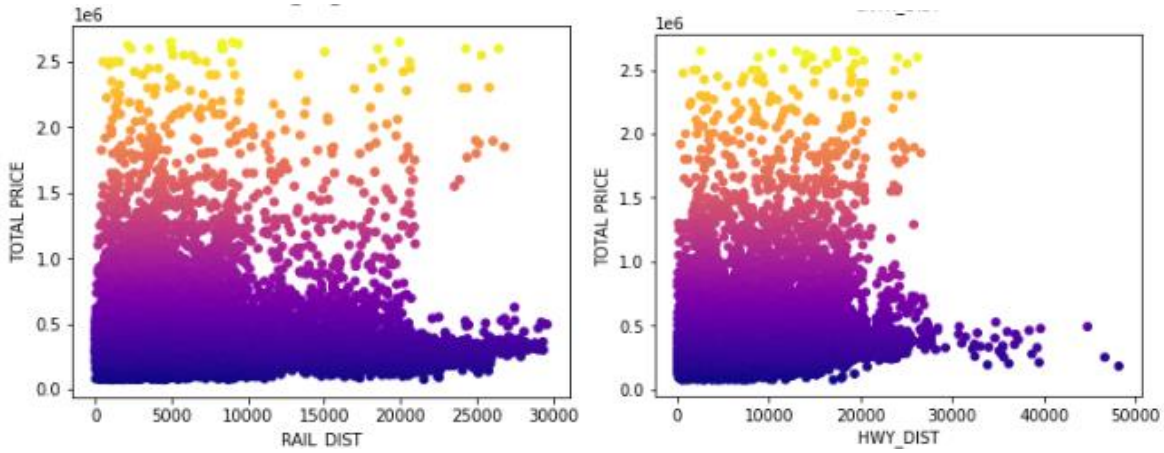
Furthermore, there is much interesting information that we can read from the above table. Based on the minimum distance data to Miami's central business district in the table above, we can see that there aren't any homes for sale in downtown Miami in the dataset. The mean of HWY_DIST and RAIL_DIST provide that the area covered in the dataset has a higher density of highways than railways. Everyone may already know about the city of Miami, however, if we just look at the OCEAN_DIST data we can see that Miami is a coastal city, with a minimum value of distance to the ocean is 236.1 square feet. According to the characteristics of coastal cities, The closer to the sea, the more expensive the house price. The fact that we can predict that the OCEAN_DIST data will have a sizable effect on the value of Miami houses. With the statistic of the distance to water in Miami City, we can see that the city of Miami has a lot of rivers, lakes, or ponds through the minimum and mean values of the data.

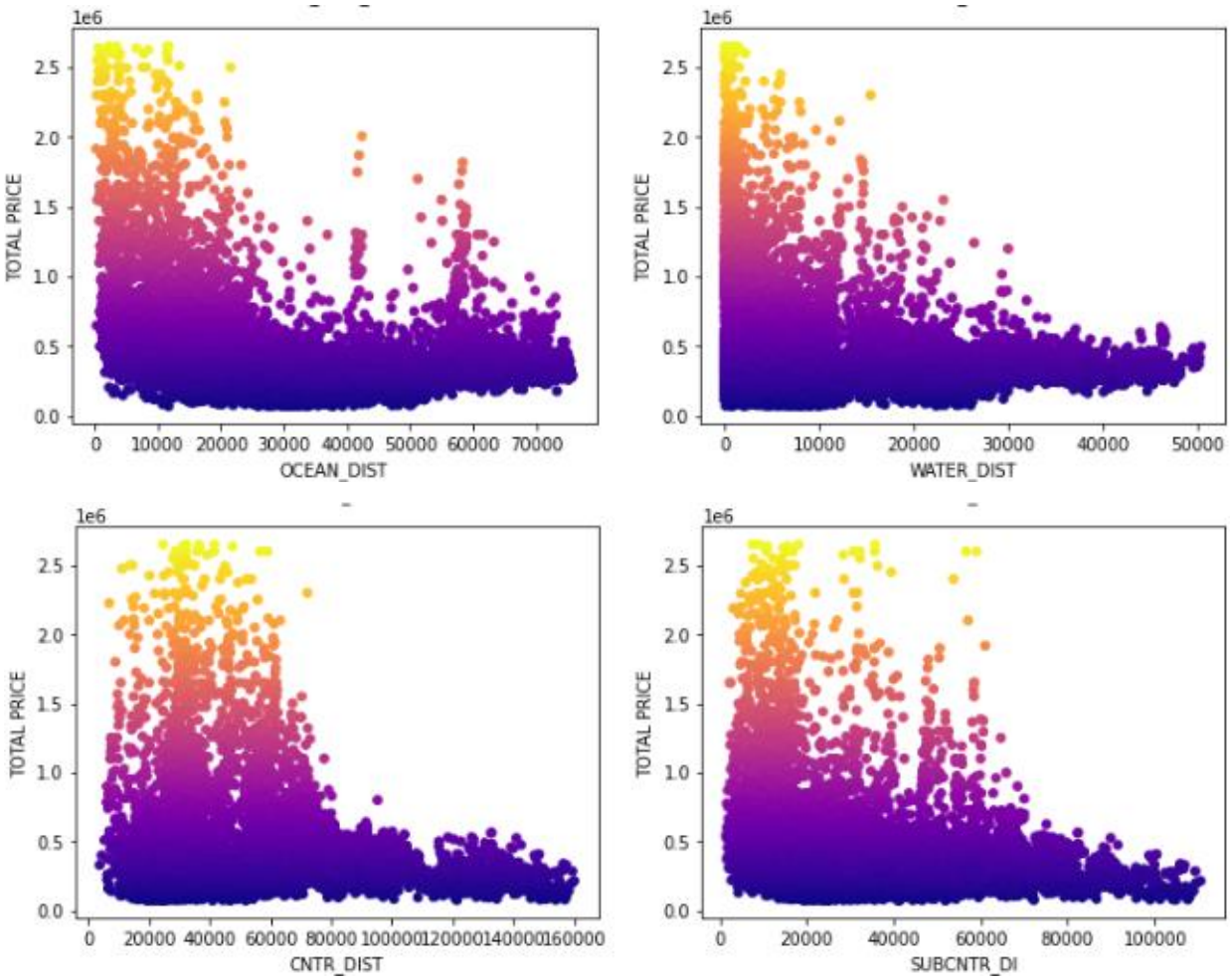
The min age of housing sold in Miami is 0 meaning that the new house was still got built and sold in Miami and the oldest that has been sold in 2016 is 96 years old however most of the houses sold in Miami have aged less than 46 years old. In 2016, The data shows that more than half of houses sold in the first half of the year. In the dataset, there is no explanation for the building structural quality data, we just know that it is in the

range from 1 to 5. However, we can look at the relationship between the building structure and its value to make a judgment about this data.

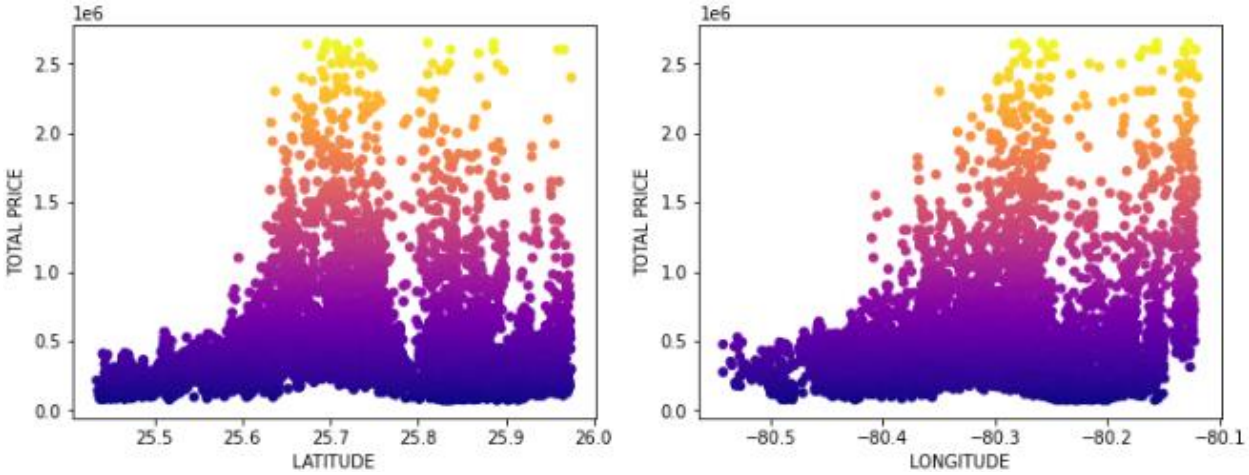


Based on the graph above, we can see that the housing price will increase when the structure quality increases from 1 to 5. The biggest difference in the maximum price is the difference between structure quality 2 and 3.

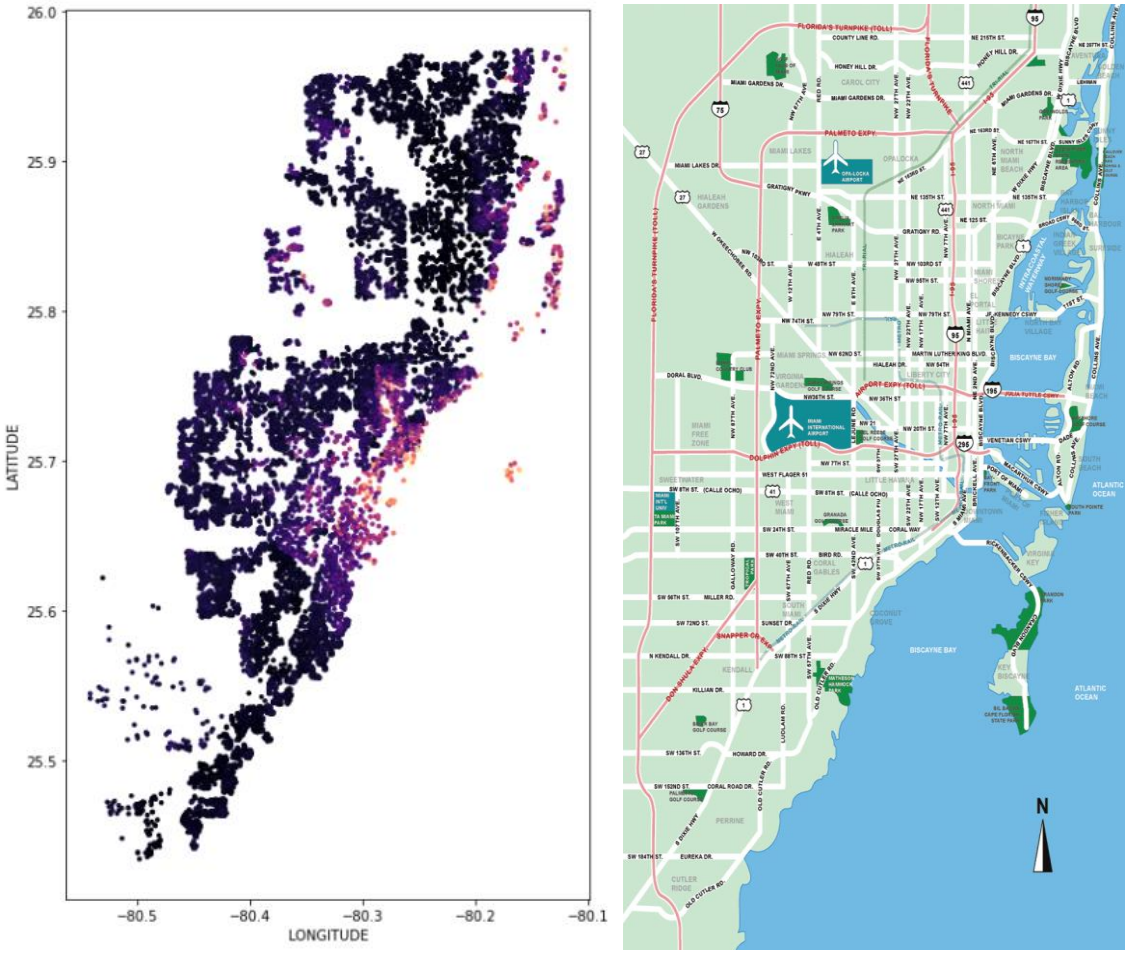




Based on the above we realize, the housing prices will rise depending on the vicinity architecture structure like sea, the highway, the mall, etc… The impacts of railway and highway distance have no discernible influence on property values in Miami, however ocean and water distance have a significant difference between properties inside 10000 square feet and those outside of 10000 square feet. The above impact is the same as the effect of and subcenter area on house price, however the evident difference in radius is 60000 square feet. The aforementioned impact is the same as the influence of the Miami central business district and nearby subcenter region on housing prices, however the evident difference in radius is 60000 square feet.



We can observe that buildings located above latitude 25.6 and west of longitude -80.4 have high values based on latitude and longitude data. Using the facts provided above, we may deduce that it is a meeting area for rivers and lakes, commercial hubs, or the beach.



Based on the map and location of buildings sold in Miami in 2016, we may conclude that the value of coastal structures, properties beyond the island, or residences near important central or rivers and lakes will be greater.

3. DATA EXPLORATION

3.1 PCA (principal component analysis)

3.1.1 Background and Goal

The purpose is to find the relative importance of the principal components and find how many PCA are necessary to explain 96% of the variance.

3.1.2 Processing

The dataset will remove PARCELNO features. After that using train_test_split to split the dataset to training and testing data with the test size is 0.3. Set the target of PCA to 96%.

3.1.3 Result

```
#show pca variance ratio
pca.explained_variance_ratio_
```

```
array([0.63717133, 0.13672547, 0.10179653, 0.07451943, 0.01600464])
```

As a result, PCA needs 5 selected components to achieve 96% result in 63.7%, 13.7%, 10.17%, 7.5%, 1.6% respectively.

```
print(pd.DataFrame(pca.components_,columns=df.drop(['PARCELNO'],axis=1).columns))
```

```
   LATITUDE     LONGITUDE  LND_SQFOOT  TOT_LVG_AREA  SPEC_FEAT_VAL  RAIL_DIST  \
0 -0.000002 -1.727214e-06   -0.017001      0.001441      -0.037704   0.080525
1  0.000005 -2.221926e-07   -0.088805     -0.009429      -0.135007   0.031802
2 -0.000002 -2.620160e-06    0.141888      0.025830       0.677053  -0.026469
3  0.000004  2.463248e-06    0.107849      0.014086       0.693224   0.073190
4  0.000003  2.967584e-06   -0.100377      0.001072       0.025767  -0.288419

   OCEAN_DIST  WATER_DIST  CNTR_DIST  SUBCNTR_DI  HWY_DIST       age  \
0    0.193463    0.161085   0.814426    0.514872  0.006555 -0.000292
1    0.880287    0.157990  -0.370668    0.188239  0.009568  0.000131
2    0.213034    0.449743   0.185687   -0.458272  0.167197 -0.000245
3    0.010438   -0.459500  -0.195525    0.493204 -0.097592  0.000008
4   -0.260160    0.362401  -0.247254    0.410360  0.691033  0.000007

     avno60plus    month_sold  structure_quality
0 -3.697770e-07  1.260083e-06          -0.000008
1  6.513319e-07 -2.166936e-06           0.000020
2 -2.402629e-07 -2.066894e-07           0.000016
3  3.630756e-07 -3.788138e-06           0.000008
4 -2.517083e-07  1.520772e-06           0.000008
```

After doing PCA on the features in the dataset, we can observe that the data analysis predictions regarding the correlation of characteristics with home prices are almost identical. OCEAN_DIST, WATER_DIST, CNTR_DIST, SUBCNTR_DI have a huge effect on the price of housing. Furthermore, in the PCA3 and PCA4 the SPEC_FEAT_VAL got 67% and 69% into the result of PCA.As a result, we may infer that the attributes OCEAN DIST, WATER DIST, CNTR DIST, SUBCNTR DI, and SPEC FEAT VAL are essential factors in determining housing values in Miami. On the contrary, the criteria that can be dropped when evaluating the target are avno60plus, month_sold and structure_quality.

3.2 Decision tree

3.2.1 Background and Goal

Divide the dataset for easier analysis. Based on the decision tree implementation, review and identify key components in the dataset.

3.2.2 Processing

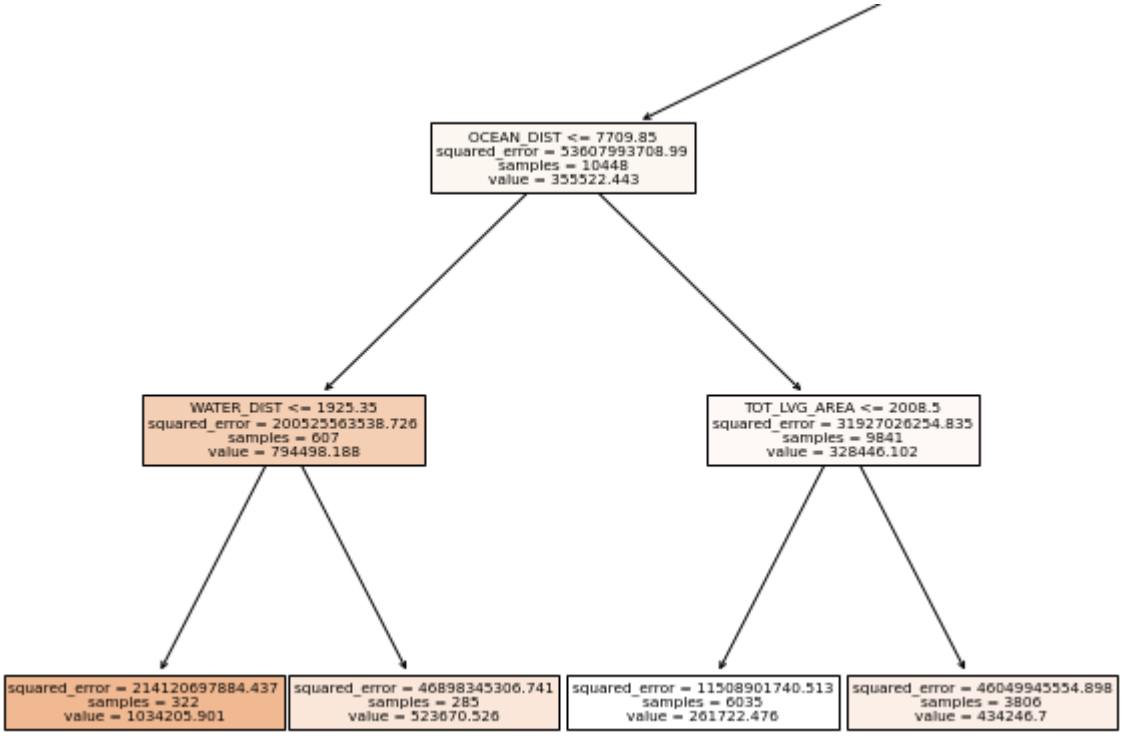
The dataset will remove PARCELNO features. After that, using train_test_split to split the dataset to training and testing data with the test size is 0.2. After completing the dataset preparation for decision tree analysis, I use DecisionTreeRegressor with a max_depth=5.

3.2.3 Data output

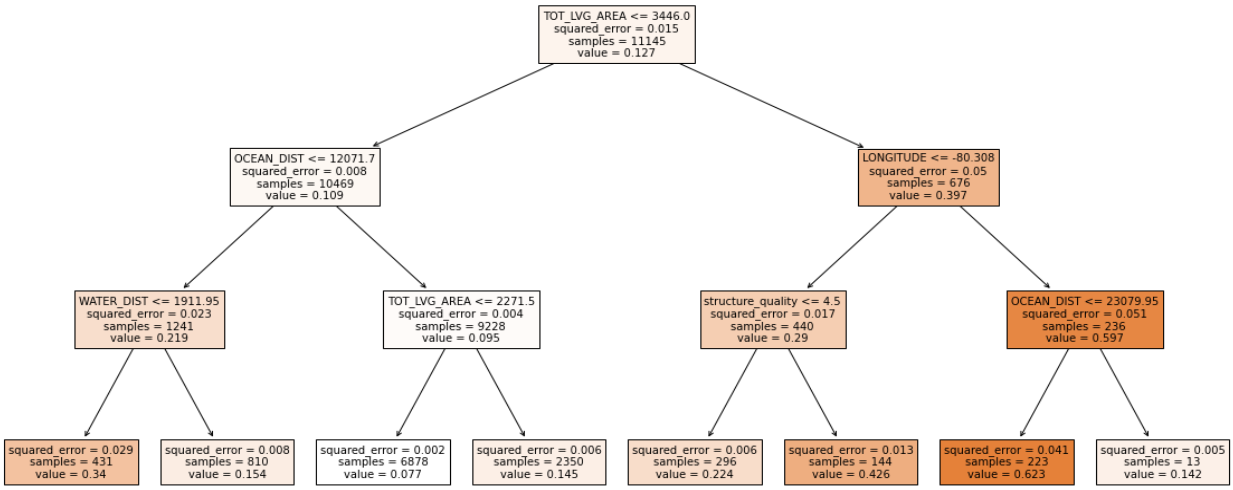
```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> 0.7437359416779608
```

After running DecisionTreeRegressor, the R-squared score is 71.36%. I experimented with visualizing by drawing a decision tree. However, because the value of the target class

is too huge, the value of square-error is a tremendous amount, causing the size of the tree to be pushed up too much and making reading the data in leaves difficult.



The solution provided is to utilize MinMaxScale to bring the value of the house price to a specified range between 0 and 1 and reduce the max_depth to 3.



AB is the value utilized as the tree's root; based on the parameter used to split the tree of the AB feature (3446 square feet), I can calculate that more than 75% of the housings

will be separated to the left of the tree. Following that,The varieties used are OCEAN_DIST, WATER_DIST, LOnd structure_quality, respectively.

4. EXPERIMENTAL METHOD

Machine Learning Hypothesis: Analyze data using machine learning methods and choose the model that best fits Miami Housing 2016 data by improving each model and comparing R-square Score.

4.1 Random Forests (and/or ExtraTrees)

4.1.1 Data preparation

The dataset will remove PARCELNO features. After that, using train_test_split to split the dataset to training and testing data with the test size is 0.2.

4.1.2 Processing

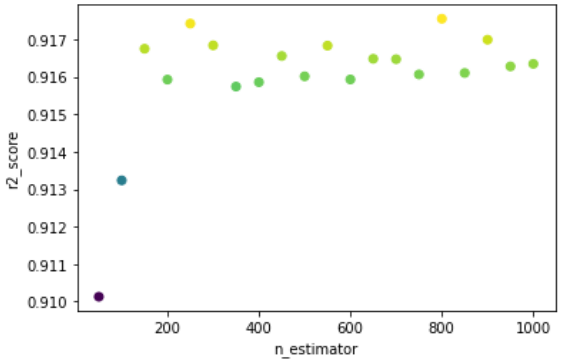
I run many ensemble methods (BaggingRegressor, ExtraTreesRegressor, RandomForestRegressor) with decision Tree regressor to check the R-square score with default setting for all of the ensemble.

Firstly, we run RandomForestRegressor and ExtraTreesRegressor with the default parameter, the R-square score is 91.2% and 90.1%, respectively.

```
<class 'sklearn.ensemble._forest.RandomForestRegressor'> 0.9017367133812813
<class 'sklearn.ensemble._forest.ExtraTreesRegressor'> 0.9125977725607441
```

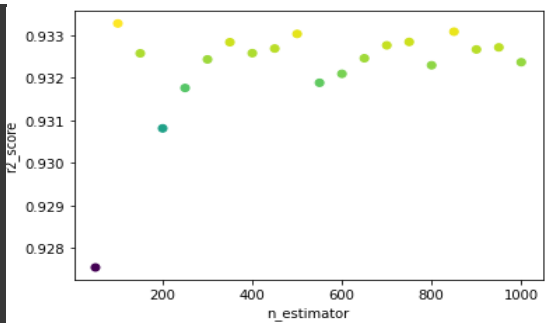
For saving time, I work separately to find the optimized max feature and number of trees in RandomForestRegressor and ExtraTreesRegressor,respectively.

```
MAX FEATURES 0.1  =>  0.9027844668633839
MAX FEATURES 0.2  =>  0.9205882969355734
MAX FEATURES 0.3  =>  0.9265765144553824
MAX FEATURES 0.4  =>  0.9265120534212997
MAX FEATURES 0.5  =>  0.9251772627779817
MAX FEATURES 0.6  =>  0.9241145298645566
MAX FEATURES 0.7  =>  0.9231733569061805
MAX FEATURES 0.8  =>  0.9202240852297965
MAX FEATURES 0.9  =>  0.9187922932490923
```



RandomForestRegressor

```
MAX FEATURES 0.1  =>  0.9312106038335468
MAX FEATURES 0.2  =>  0.9309176977645657
MAX FEATURES 0.3  =>  0.9330360290724798
MAX FEATURES 0.4  =>  0.9326493196091176
MAX FEATURES 0.5  =>  0.9320483876590214
MAX FEATURES 0.6  =>  0.9326295121379262
MAX FEATURES 0.7  =>  0.9309755158625448
MAX FEATURES 0.8  =>  0.9307463543441267
MAX FEATURES 0.9  =>  0.9321258199973652
```



ExtraTreesRegressor

After running the aforementioned two parameters with the RandomForestRegressor , we see that the results of the tests do not change much, therefore I will use max feature from 0.1 to 0.4 and sqrt (best result after testing) and n_estimator form 200 to 800(Best range after testing) for the Randomized Search Cross Validation.

```
rf_random.best_params_
```

```
{'n_estimators': 666,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 0.2,
 'max_depth': 70,
 'bootstrap': False}
```

```
R-square score =>  0.9297940849133655
```

After running the Randomized Search I got the best combination of the Random forests regression. I ran the

Random forest regression again to check the improvement.

As we can see, the R-square score slightly increases from 90.1% with the default parameter to 93% after we use the parameter combination result of RandomizedSearchCV.

```
{'n_estimators': 266,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 0.2,
 'max_depth': None,
 'bootstrap': False}
```

```
ExtraTreesRegressor R-square score =>  0.914262413918588
```

We work the same process with the ExtraTreesRegressor:

R-square score after running RandomizedSearchCV is 91,5% which is an increase by 0.3% from the output before RandomizedSearchCV.

Based on the data obtained after adjusting the Hyperparameters, we can observe that the R-square score rises slightly, only by around 4%. Although the R-square score increases, the operation time is substantially increased due to the large number of trees in the algorithms. Therefore, if you decide to use optical data, the user must consider the key of time.

```
RandomForestRegressor Accuracy 0  :  0.9184156011281825
RandomForestRegressor Accuracy 1  :  0.9147297610804593
RandomForestRegressor Accuracy 2  :  0.9119745331278717
RandomForestRegressor Accuracy 3  :  0.9134047110886041
RandomForestRegressor Accuracy 4  :  0.910119222632638
RandomForestRegressor Accuracy 5  :  0.9137252427128376
RandomForestRegressor Accuracy 6  :  0.9216285333629888
RandomForestRegressor Accuracy 7  :  0.9116555840338306
RandomForestRegressor Accuracy 8  :  0.9060580847964467
RandomForestRegressor Accuracy 9  :  0.9176321314943419
```

```
ExtraTreesRegressor Accuracy 0  :  0.9188440212694383
ExtraTreesRegressor Accuracy 1  :  0.9152465364776504
ExtraTreesRegressor Accuracy 2  :  0.9103174836196166
ExtraTreesRegressor Accuracy 3  :  0.9139768853561332
ExtraTreesRegressor Accuracy 4  :  0.9111615092316913
ExtraTreesRegressor Accuracy 5  :  0.9152260175032683
ExtraTreesRegressor Accuracy 6  :  0.9218924693952855
ExtraTreesRegressor Accuracy 7  :  0.9108155878151123
ExtraTreesRegressor Accuracy 8  :  0.9057897914104251
ExtraTreesRegressor Accuracy 9  :  0.9183384407454712
```

I have tried running with cross validation (n_split = 10) with optimal parameter, the result is not too obvious. On the side of RandomForestRegressor, we see the growth is not too much, while ExtraTreesRegressor gives close results is similar to default parameter even lower

4.2 AdaBoost

4.2.1 Data preparation

The dataset will remove PARCELNO features. After that, using train_test_split to split the dataset to training and testing data with the test size is 0.2.

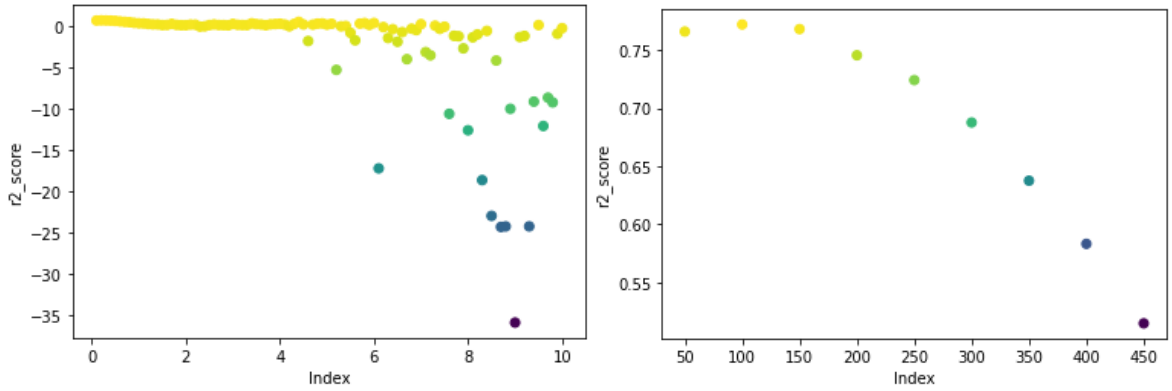
4.2.2 Processing

Firstly, I run adaBoostRegressor with the default parameter. The R-square score is 39.94%.

```
AdaBoost Regressor R2 Score :  0.3994335936182781
```

To save time for turning Hyperparameter, we test each parameter of ADABOOST separately with n_estimator in range 50 to 1000 and learning rate from 0.1 to 10 to identify

the optimal learning rate and number of trees (n_estimator_) and 3 loss parameter (linear, square and exponential).



The R-square score slightly decreases when the learning rate increases over 1. Therefore we run the code again to find the optimal learning rate from 0.1 to 1.

```
learning_rate:  0.1 n_estimators : 50  :  0.7576289407789776
learning_rate:  0.2 n_estimators : 50  :  0.7817574281979613
learning_rate:  0.3 n_estimators : 50  :  0.7575798579085592
learning_rate:  0.4 n_estimators : 50  :  0.737301943220416
learning_rate:  0.5 n_estimators : 50  :  0.6734554366009953
learning_rate:  0.6 n_estimators : 50  :  0.6413079026835256
learning_rate:  0.7 n_estimators : 50  :  0.5699248966842003
learning_rate:  0.8 n_estimators : 50  :  0.5214360454645036
learning_rate:  0.9 n_estimators : 50  :  0.445808904427641
learning_rate:  1.0 n_estimators : 50  :  0.3874566693178324
 MAX VALUE
learning_rate:  0.2 n_estimators : 50  =>  0.7817574281979613
```

For the number of trees, the graph above shows that the more trees we have the less the R-square score.

After running and testing with parameters , I decided to work with n_estimator = 100 and learning_rate =0.2. Additionally, I keep trying to find a loss function to use when updating the weights after each boosting iteration.

```
AdaBoost Regressor R2 Score  with loss = linear:  0.7255442303416724
AdaBoost Regressor R2 Score  with loss = square:  0.7034784502505212
AdaBoost Regressor R2 Score  with loss = exponential:  0.7531084686883385
```

As we can see, the square loss function is the worthiest function used in this dataset, while the exponential function is the best, therefore I decided to perform the Cross Validation(ShuffleSplit, n_split =10, test_size=0.3) using the exponential function to double-check the technique that we built with the adaBoostRegressor.

```
Accuracy 0  :  0.7759534656446121
Accuracy 1  :  0.7500551570851755
Accuracy 2  :  0.7386732305616892
Accuracy 3  :  0.7495217786674446
Accuracy 4  :  0.7636302824759931
Accuracy 5  :  0.7601466992432311
Accuracy 6  :  0.7524913216761154
Accuracy 7  :  0.7457538021518542
Accuracy 8  :  0.7575286321682282
Accuracy 9  :  0.7546729954664895
```

As we can see, the final result that we got when running the Cross Validation is around 76% which is a huge increase with the default parameter setting that we got before. Therefore, with the optimal of the parameter(n_estimator =100, learning_rate =0.2 and loss function = exponential) I got the R-Square increase by 37% from 39.9% to 77.4 %.

4.3 GradientBoost

4.3.1 Data preparation

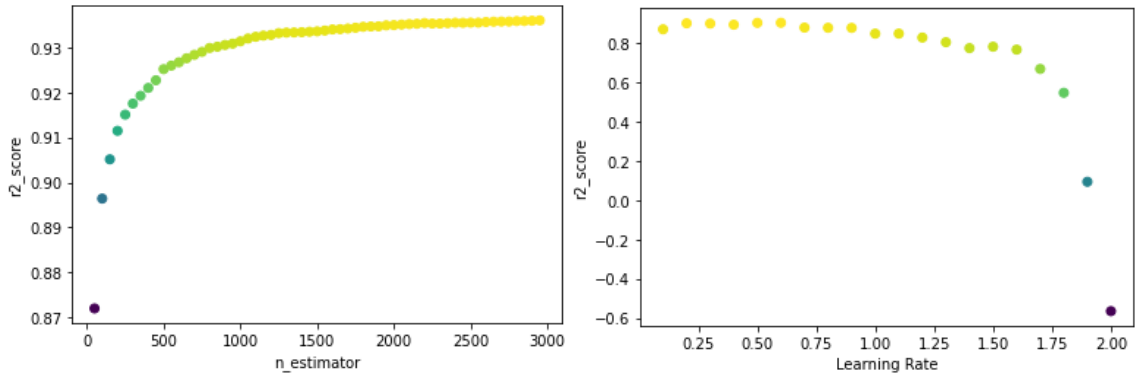
4.3.2 Processing

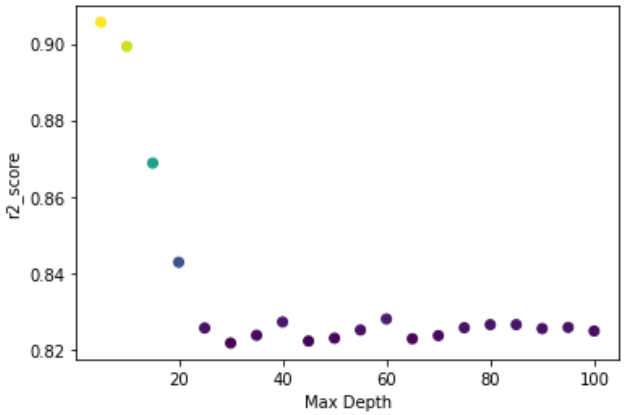
The dataset will remove PARCELNO features. After that, using train_test_split to split the dataset to training and testing data with the test size is 0.2.

Firstly, we run GradientBoostRegressor with the default parameter. The R-square score is 88.8%.

```
GradientBoostingRegressor R-suare score : 0.888233
```

To save time for turning Hyperparameter, we test each parameter of GradientBoostRegressor separately with n_estimator in range 50 to 3000 and learning rate from 0.1 to 2 to identify the optimal learning rate and number of trees (n_estimator_).





Based on the graphs above, we know that the higher the n_estimator the greater R-square score is, however, I am concerned about the time running if the n_estimator is too high, therefore I choose the n_estimator around 1000. Furthermore,if the Learning rate is over 1, the R-square score dropdown.Due to the fact that I decide to choose the n_estimator = 1000 and 1500, learning_rate = 0.3 and 0.4 for the Randomized Search Cross Validation.

```
{'n_estimators': 1500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 0.4,
 'max_depth': 5}
```

```
GradientBoostingRegressor R-suare score : 0.912374
```

The output rises a bit to 91.2% from 88.82%. However, when the experiment is repeated with Cross Validation, the output is only approximately 88%. With n_estimator set to 1500, the value obtained is not so nice (raising running time concerns).

4.4 Multi-layer Perceptron

4.4.1 Data preparation

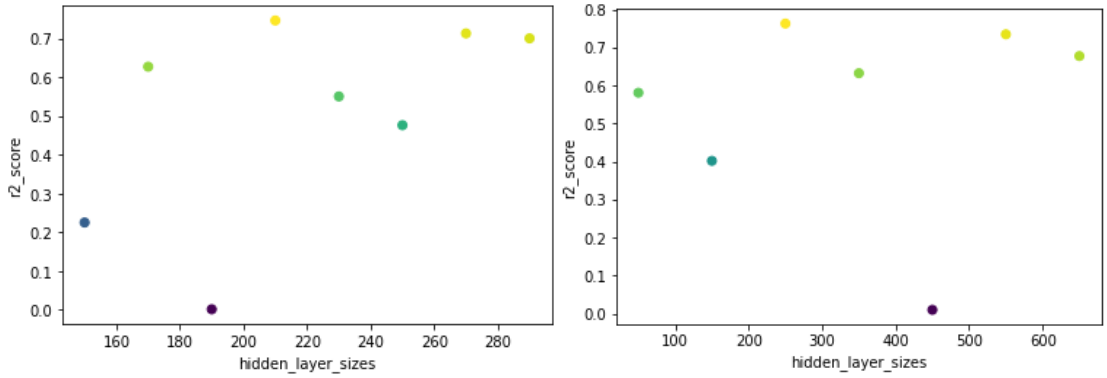
The dataset will remove PARCELNO features. After that, using train_test_split to split the dataset to training and testing data with the test size is 0.2.

4.4.2 Processing

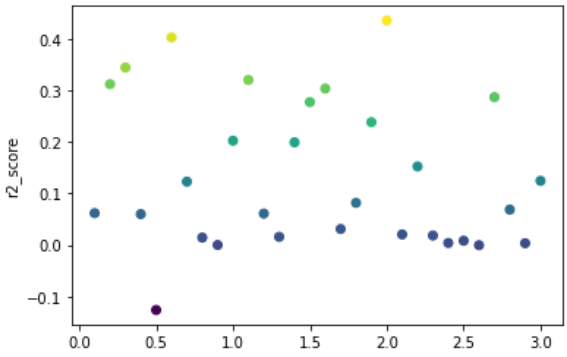
At the beginning, we run the MLPRegressor with default parameter and get the result is 81.9%. Then we run the loop to test the optimal hidden_layer_sizes and learning_rate.

```
MLPRegressor R-square Score: => 0.8192060037124395
```

The reason for this is because I want to reduce the range of the parameters to reduce the running time of turning Hyperparameters.



After running with different hidden_layer_size, I discovered that the results do not adhere to any norms, but the results between 200 and 300 are fairly acceptable, therefore I chose to utilize the hidden_layer_size from 200 to 300 for turning Hyperparameters.




```
Best learning_rate_init:  2.0  => 0.43601539031918113
```

For the Learning Rate, I ran twice and both of the results have the best learning_rate_init at 2.0, I keep doing that with the solver (lbfgs, sgd, adam)and got the best output with lbfgs. After completing the selection of parameters, we run tuning Hyperparameters with RandomSearchCV.

```
{'max_iter': 1000,
 'learning_rate_init': 0.6,
 'hidden_layer_sizes': 230,
 'alpha': 0.01}
```

```
MLPRegressor R-square Scpre: => 0.853661072544327
```

With MLP, the results achieved after tuning Hyperparameters are pretty good, with R-square increasing from 81.9% to 85.4% and the completion time being relatively constant.

5. RESULTS AND ANALYSIS

Overall, after running and optimizing those machine learning algorithms, I have the table below.

	Random Forests Regression	**Extra Trees Regression**	**ADABOOST Regression**	**Gradient Boost Regression**	**MLP**
R_Square before optimizing	90.17%	91.25%	39.94%	88.8%	81.9%
R_Square after optimizing	92.8%	91.42%	77.5%	91.2%	85.4%

The table illustrates the output of those algorithms with the Miami Housing 2016 dataset. The best results we achieved with the R-Square score were Random Forest Regression with 92.8% followed by Extra Trees Regression and Gradient Boost Regression with 91.42% and 91.2% respectively. Although ADABOOST Regression was the lowest score, the optimized parameter result was the most successful with a 37% increase in R-square score. In terms of running time of algorithms, Random Forests Regression and Extra Trees Regression are more noticeable than Gradient Boost Regression because the time consuming of Gradient Boost Regression is quite large compared to the other two algorithms. Multi-layer Perceptron is still an interesting algorithm, I did not fully optimise MLP in this example because increasing the number and range of parameters increases the running time.

REFERENCES

(1) Chaudhury, S. (2020, August 31). Tuning of adaboost with computational complexity. Medium. Retrieved November 28, 2022, from https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20

(2) Gaurav, M. (2019, December 17). How to find the optimum number of hidden layers and nodes in a neural network model? Retrieved November 27, 2022, from https://datagraphi.com/blog/post/2019/12/17/how-to-find-the-optimum-number-of-hidden-layers-and-nodes-in-a-neural-network-model

(3) Koehrsen, W. (2018, January 10). *Hyperparameter tuning the random forest in python*. Medium. Retrieved November 30, 2022, from https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74