

Classifying Houses into Different Price Range using Neural Network and Tree Classifiers from American Housing Survey 2017 Dataset

1st Darp Raithatha
Masters in Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey
draithat@stevens.edu

2nd Likith Nandigam
Masters in Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey
lnandiga@stevens.edu

3rd Monami Mukhopadhyay
Masters in Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey
mmukhopa@stevens.edu

I. PROBLEM STATEMENT

The main goal of this project is to predict the range of selling price of house with a high degree of predictive accuracy using various Machine Learning methods. Given house sale data or explanatory variable such as number of bedrooms, number of bathrooms in unit, housing cost, annual commuting cost etc, we build our model. Next, the model is evaluated with respect to test data, and plot the prediction and coefficients. In this way, the value of a house can be calculated in a given geographical area.

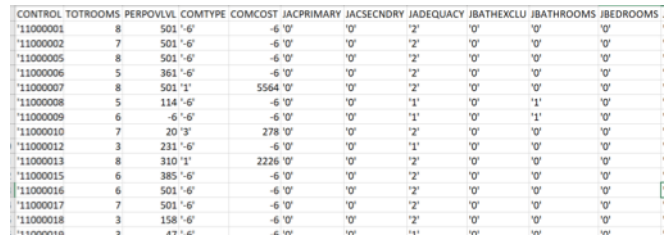
II. DATA

We are using American Housing Survey 2017 (household.csv in AHS 2017 National PUF v3.0 CSV.zip). Since the dataset is very big, we have uploaded it on Google Drive. It could not be uploaded in github repo. There is another .csv file that consist of the mapping information of each feature name to their actual meaning and data type information. This file is already present on github repo. In the AHS microdata, the basic unit is an individual housing unit. Each record shows most of the information associated with a specific housing unit or individual, except for data items that could be used to personally identify that housing unit or individual. Our dataset comprises of housing data features like TOTROOMS (Number of rooms in unit), PERPOVLVL (Household income as percent of poverty threshold (rounded)), COMCOST (Total annual commuting cost), JBATHROOMS (Number of bathrooms in unit), UNITSF (Square footage of unit), JGARAGE (Flag indicating unit has a garage or carport), JFIREPLACE (Flag indicating unit has a useable fireplace) etc., and target column as MARKETVAL (Current market value of unit) to evaluate model and also check which amongst all features is the most correlated feature for price prediction. Two primary datasets used are:

- Household.csv in AHS 2017 National PUF v3.0 CSV.zip
- AHSDICT 15NOV19 21 17 31 97 S

A. Prerequisites

- Python 3.7.0
- Numpy
- Pandas
- Scipy
- Scikit-learn
- Matplotlib



	CONTROL	TOTROOMS	PERPOVLVL	COMTYPE	COMCOST	JACPRIMARY	JACSECNDRY	JADEQUACY	JBATHCLU	JBATHROOMS	JBEDROOMS
'110000001	8	501	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000002	7	501	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000005	8	501	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000006	5	361	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000007	8	501	'1'		5564 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000008	5	114	'6'		-6 '0'	'0'	'1'	'0'	'1'	'0'	'0'
'110000009	6	-6	'6'		-6 '0'	'0'	'1'	'0'	'1'	'0'	'0'
'110000010	7	20	'3'		278 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000012	3	231	'6'		-6 '0'	'0'	'1'	'0'	'0'	'0'	'0'
'110000013	8	310	'1'		2226 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000015	6	385	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000016	6	501	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000017	7	501	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000018	3	158	'6'		-6 '0'	'0'	'2'	'0'	'0'	'0'	'0'
'110000019	9	477	'6'		-6 '0'	'0'	'141	'0'	'0'	'0'	'0'

Fig. 1. Screenshot of Dataset

```
data['MARKETVAL'].describe()

count    3.995100e+04
mean     3.382110e+05
std      5.246493e+05
min      1.000000e+00
25%      1.210225e+05
50%      2.265120e+05
75%      3.958310e+05
max      9.999998e+06
Name: MARKETVAL, dtype: float64
```

Fig. 2. Description of the Dataset

III. PROJECT IMPLEMENTATION

We defined machine learning models such as Random Forest Classifier, k-Nearest Neighbor Classifier, Decision Tree Classifier, Multi-layer Perceptron Classifier, AdaBoost, and, Ensemble Method (Stacking) using most of the explanatory variables describing every aspect of residential homes and predict the price range of each home.

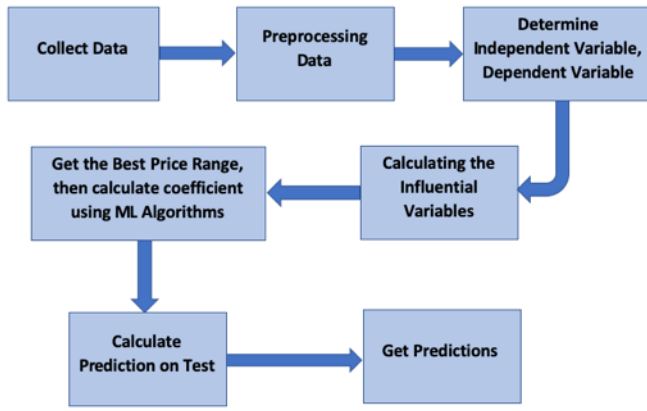


Fig. 3. Project Implementation - Flowchart

IV. DATA CLEANING

The dataset was cleaned to make it free from erroneous or irrelevant data. By filling up missing values, removing rows, and reducing data size, the final dataset was (36358 rows X 526 columns). All the features were evaluated, and the results were used to reduce the dimensionality of the dataset and to check which amongst all features is the most correlated feature for price prediction. Finally, the data was critically analyzed for the distribution of data, and derived relevant statistics. Then the data was checked for inconsistencies and removed accordingly.

	TOTROOMS	PERPOVLVL	COMCOST	DINING	LAUNDY	STORIES	COMDAYS	DIST	RATINGS	RATINGNH	CELLPHONE	WEIGHT
TOTROOMS	1	0.2	0.021	0.58	0.24	0.1	-0.095	0.022	0.12	0.11	0.21	-0.036
PERPOVLVL	0.2	1	0.024	0.089	0.087	0.13	-0.0015	0.027	0.05	0.049	0.16	-0.054
COMCOST	0.021	0.024	1	0.012	0.0072	-0.0097	0.086	0.89	0.0012	0.0014	0.011	-0.023
DINING	0.58	0.089	0.012	1	0.045	0.081	-0.013	0.016	0.063	0.057	0.078	-0.051
LAUNDY	0.24	0.087	0.0072	0.045	1	-0.045	-0.013	0.016	0.068	0.053	0.063	0.056
STORIES	0.1	0.13	-0.0097	0.061	-0.045	1	-0.016	-0.016	0.024	0.033	0.025	-0.028
COMDAYS	-0.035	-0.0015	0.086	-0.013	-0.016	-0.016	1	-0.0059	-0.029	-0.018	0.013	-0.0098
DIST	0.022	0.027	0.89	0.016	0.016	-0.016	-0.0059	1	-0.0027	0.0052	0.01	-0.019
RATINGS	0.12	0.05	0.0012	0.063	0.068	0.024	-0.029	-0.0027	1	0.61	0.018	0.022
RATINGNH	0.11	0.049	0.0014	0.057	0.053	0.033	-0.018	0.0052	0.61	1	0.01	0.027
CELLPHONE	0.21	0.16	0.011	0.078	0.063	0.025	0.013	0.01	0.018	0.01	1	-0.03
WEIGHT	-0.036	-0.054	-0.023	-0.051	0.056	-0.028	-0.0098	-0.019	0.022	0.027	-0.03	1

Fig. 4. Correlation Matrix

V. FEATURE ENCODING

To predict the price range for houses, we divided the entire range of MARKETVAL values into smaller ranges and encoded them into set of classes as [1, 2, 3, 4, 5, 6, 7, 13]. The function used for the same is as in Fig. 5.

The distribution of training and testing data based on the above price range set of classes can be visualized as in Fig. 6 and Fig. 7.

We also plotted a bar graph representing the top 10 features based on their importance in determining the house price range:

- PROTAXAMT - Monthly Property Tax Amount
- TOTHCAMT - Monthly Total Housing Cost

```

# Function to assign code for different price ranges
def price_range(price):
    if price < 100000:
        return 1
    elif price >= 100000 and price < 250000:
        return 2
    elif price >= 250000 and price < 500000:
        return 3
    elif price >= 500000 and price < 750000:
        return 4
    elif price >= 750000 and price < 1000000:
        return 5
    elif price >= 1000000 and price < 1250000:
        return 6
    elif price >= 1250000:
        return 7
    else:
        print(price)
        return 13
  
```

Fig. 5. Feature Encoding

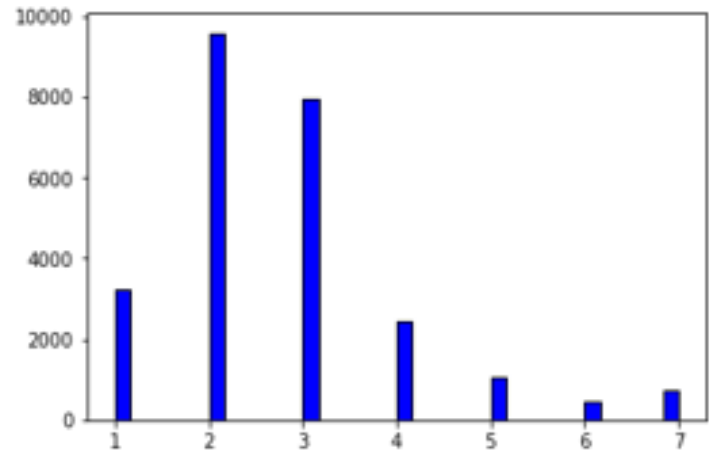


Fig. 6. Train

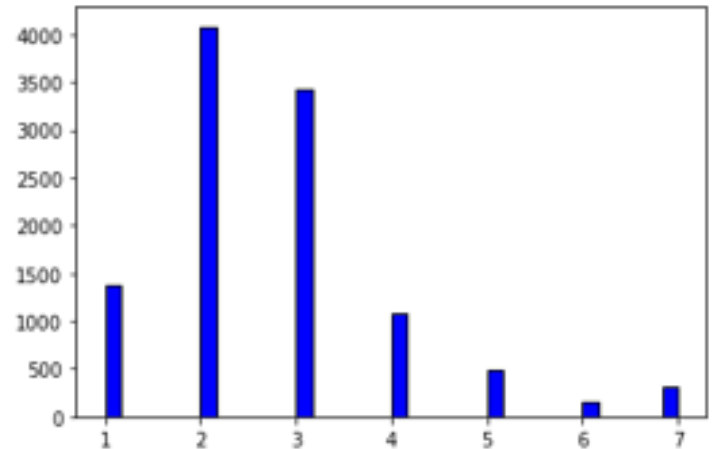


Fig. 7. Test

- INSURAMT - Monthly Homeowner or Renter Insurance Amount
- MORTAMT - Monthly Total Mortgage Amount (all mortgages)
- TOTBALAMT - Total Remaining Debt Across all Mortgages or Similar Debts for this Unit
- DIVISION - Census Division
- UNITSIZE - Unit Size (Square Feet)
- BATHROOMS - Number of Bathrooms in Unit
- FINCP - Family Income (Past 12 Months)
- HINCP - Household Income (Past 12 Months)

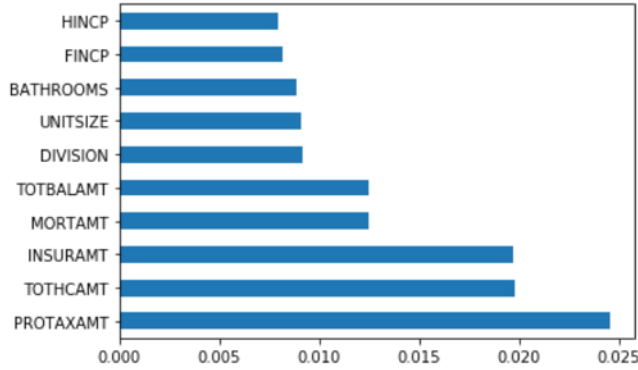


Fig. 8. Top 10 features for determining house price range

VI. ALGORITHMS IMPLEMENTED

In this project, our aim was to implement algorithms which will be able to learn and classify the new observations to correct house price ranges. So, we decided to use below machine learning algorithms for the same-

- Random Forest (RandomForestClassifier)
- K-Nearest Neighbor (KNeighborsClassifier)
- Decision Tree (DecisionTreeClassifier)
- Multi-layer Perceptron Classifier (MLPClassifier)
- AdaBoost Classifier (AdaBoostClassifier)
- Ensemble Method (Stacking)

A. Random Forest (RandomForestClassifier)

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The predictive performance of Random Forest Classifier can compete with the best supervised learning algorithms. It provides a reliable feature importance estimate. It also offers efficient estimates of the test error without incurring the cost of repeated model training associated with cross-validation.

In order to increase test accuracy, we tried executing the algorithm using n estimators = 100, 200, 500, 1000. We finally retained the best result that we could get.

The accuracies obtained (in percentage) using RandomForestClassifier:

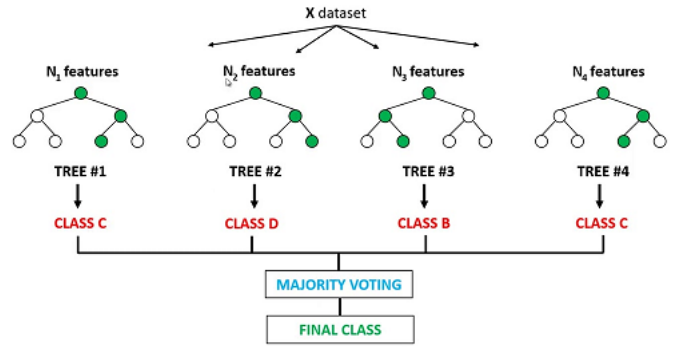


Fig. 9. Random Forest Classification

```
# Using Random Classifier
model = train_model(X_train, y_train, RandomForestClassifier, n_estimators=200, random_state=20, verbose=True)
accuracy(X_train, X_test, y_train, y_test, model)
# Top 10 features that determine price
pd.Series(model.feature_importances_, x.columns).sort_values(ascending=True).nlargest(10).plot.barh(align='center')
```

Fig. 10. Random Forest

- Training Accuracy – 100.00
- Testing Accuracy – 59.49

B. k-Nearest Neighbors

KNN or k-nearest neighbours is the simplest classification algorithm. This classification algorithm does not depend on the structure of the data. Whenever a new example is encountered, its k nearest neighbours from the training data are examined. Distance between two examples can be the euclidean distance between their feature vectors. The majority class among the k nearest neighbours is taken to be the class for the encountered example. This is a fairly simple algorithm which is easy to explain and understand/interpret. It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training. We can also get high accuracy

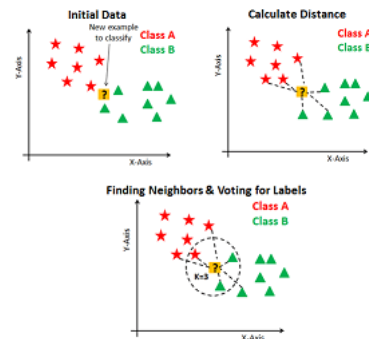


Fig. 11. k-Nearest Neighbors Classification

For the kNN algorithm, we modified the value for n neighbors = 3, 4, 5, 6, 7, 8, 9, 10 and finally retained the value that would give us best test accuracy.

```
# Using kNN Classifier
model = train_model(X_train, y_train, KNeighborsClassifier, n_neighbors=7)
accuracy(X_train, X_test, y_train, y_test, model)
```

Fig. 12. k-Nearest Neighbors

The accuracies obtained (in percentage) using k-Nearest Neighbors:

- Training Accuracy – 60.19
- Testing Accuracy – 47.36

C. Decision Tree

Decision Tree Classifier is a simple and widely used classification technique. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached. Compared to other algorithms, decision trees requires less effort for data preparation during pre-processing. A decision tree does not require normalization of data. Also, decision tree does not require scaling of data as well.

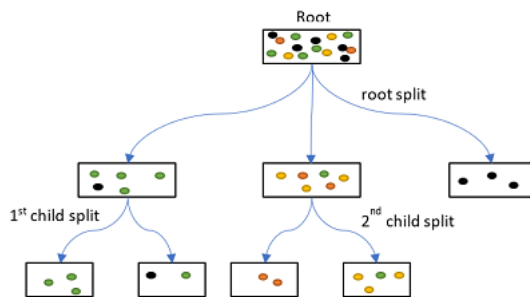


Fig. 13. Decision Tree Classification

We changed the values for max depth = 2, 3, 4, 5, 6, 7, 8, 9, 10, 20 to finally select the best test accuracy result for Decision Tree.

```
# Using Decision Tree Classifier
model = train_model(X_train, y_train, DecisionTreeClassifier, max_depth=8)
accuracy(X_train, X_test, y_train, y_test, model)
```

Fig. 14. Decision Tree

The accuracies obtained (in percentage) using Decision Tree:

- Training Accuracy – 65.08
- Testing Accuracy – 60.60

D. Multi-layer Perceptron

MLPClassifier relies on an underlying Neural Network to perform the task of classification. A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs.

We tried different values for alpha, max iter and solver to get the best accuracy results.

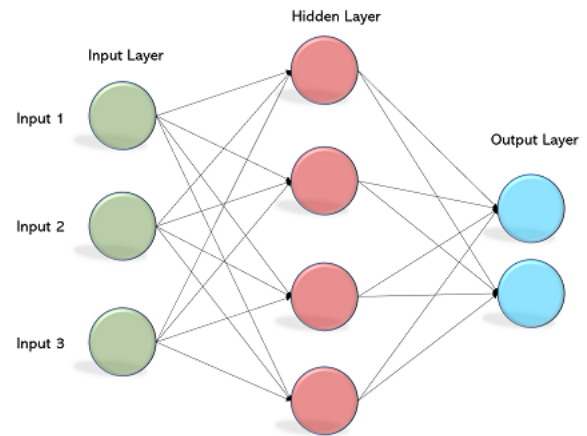


Fig. 15. Multi-layer Perceptron Classification

```
Multi-layer Perceptron Classifier
M mlp = MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=0.0001)
mip.fit(X_train, y_train)
accuracy(X_train, X_test, y_train, y_test, mip)
```

Fig. 16. Multi-layer Perceptron

The accuracies obtained (in percentage) using Multi-layer Perceptron:

- Training Accuracy – 37.65
- Testing Accuracy – 37.40

E. AdaBoost

Ada-boost classifier combines weak classifier algorithm to form strong classifier. A single algorithm may classify the objects poorly. But if we combine multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, we can have good accuracy score for overall classifier. Boosting algorithms are less affected by the overfitting problem. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier.

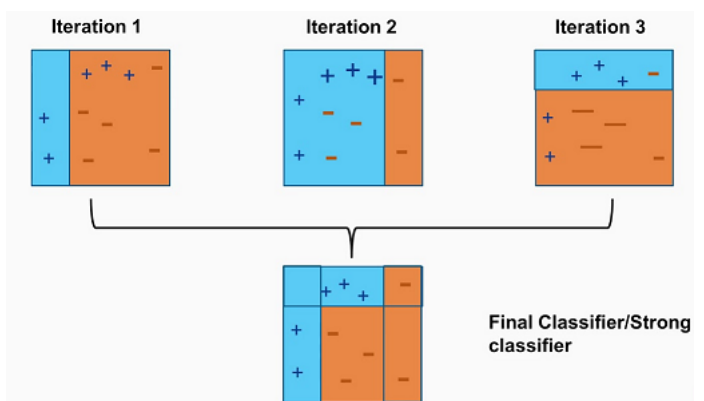


Fig. 17. AdaBoost Classification

For AdaBoostClassifier, we used n estimators = 20, 50, 70, 100, 300, 500, 1000, max depth = 3, 4, 5, 7, 8, 20, 15, 20,

learning rate = 0.1, 0.3, 0.5, 0.01, 0.08, 0.003. Finally, we got below best accuracy values.

```
abc = AdaBoostClassifier(n_estimators=50, base_estimator=DecisionTreeClassifier(max_depth=8), learning_rate=0.01)
abc = abc.fit(X_train, y_train)
accuracy(X_train, X_test, y_train, y_test, abc)
```

Fig. 18. AdaBoost

The accuracies obtained (in percentage) using AdaBoost:

- Training Accuracy – 73.54
- Testing Accuracy – 61.29

F. Ensemble Method (Stacking)

Stacking is an ensemble learning technique that combines multiple classification via a meta-classifier. The base level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features. The objective is to increase model accuracy.

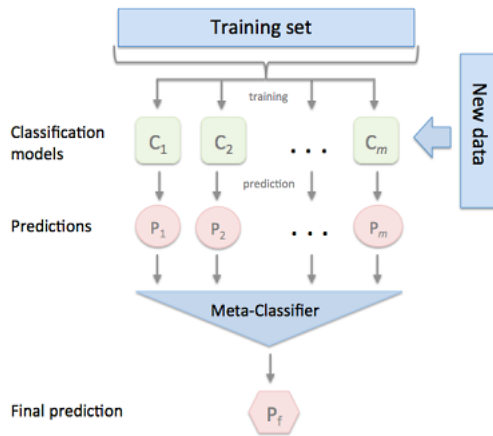


Fig. 19. Ensemble Method (Stacking) Classification

```
# Decision Tree Classifier
model1 = DecisionTreeClassifier()
model1.fit(X_train_v, y_train_v)
val_pred1=model1.predict(X_val)
test_pred1=model1.predict(X_test)
val_pred1=pd.DataFrame(val_pred1)
test_pred1=pd.DataFrame(test_pred1)

# Random Forest Classifier
model2 = RandomForestClassifier()
model2.fit(X_train_v, y_train_v)
val_pred2=model2.predict(X_val)
test_pred2=model2.predict(X_test)
val_pred2=pd.DataFrame(val_pred2)
test_pred2=pd.DataFrame(test_pred2)
```

Fig. 20. Ensemble Method (Stacking) - 1

The accuracies obtained (in percentage) using Ensemble Method (Stacking):

- Testing Accuracy – 43.43

```
# MLP Classifier
model3 = mlp
model3.fit(X_train_v, y_train_v)
val_pred3=model3.predict(X_val)
test_pred3=model3.predict(X_test)
val_pred3=pd.DataFrame(val_pred3)
test_pred3=pd.DataFrame(test_pred3)

# k-Nearest Neighbor Classifier
model4 = KNeighborsClassifier()
model4.fit(X_train_v, y_train_v)
val_pred4=model4.predict(X_val)
test_pred4=model4.predict(X_test)
val_pred4=pd.DataFrame(val_pred4)
test_pred4=pd.DataFrame(test_pred4)
```

Fig. 21. Ensemble Method (Stacking) - 2

```
df_val=pd.concat([X_val, val_pred1.reindex_like(
df_test=pd.concat([X_test, test_pred1.reindex_li

df_val.dropna(axis='columns', inplace=True)
df_test.dropna(axis='columns', inplace=True)

# Getting the test accuracy score for Stacking
model = LogisticRegression()
model.fit(df_val, y_val)
model.score(df_test, y_test)
```

Fig. 22. Ensemble Method (Stacking) - 3

VII. CONCLUSIONS

A range of different models were explored, from relatively simple to more complex ones, with the goal of best predicting home price ranges but also utilizing different data science skills in the process. For this particular problem, the algorithm with best accuracy value is AdaBoost Classifier with test accuracy score of 0.6129 and therefore it can be considered as a good classifier algorithm for house price range prediction problem. Also, the Decision Tree Classifier is close enough with 0.6060 accuracy score. We have tried tuning each algorithm with different hyper-parameter values and finally kept the best results for each.

	Test Accuracy
Random Forest Classifier	59.49
k-Nearest Neighbor Classifier	47.36
Decision Tree Classifier	60.6
Multi-layer Perceptron Classifier	37.4
AdaBoost Classifier	61.29
Ensemble Method (Stacking)	43.43

Fig. 23. Test Accuracy Summary

VIII. CONTRIBUTION

Each of the members had unique contributions towards the project. While three of us participated equally in Data

Cleaning process, we worked on different ML algorithms as below:

- Darp Raithatha is responsible for preliminary research, gathering information, correlation determination, external materials to refer to, model evaluation and code testing. He worked on AdaBoost Classifier and Ensemble Method (Stacking) algorithms.
- Likith Nandigam is responsible for cleaning the dataset, compiling the code, code testing, result extraction, and documentation. He worked on Random Forest Classifier and k-Nearest Neighbor Classifier algorithms.
- Monami Mukhopadhyay is responsible for code testing, result extraction, project compilation, documentation, gathering and implementing references. She worked on Decision Tree Classifier and Multi-layer Perceptron Classifier algorithms.

IX. CODE EXECUTION DETAILS

For our project, we applied six different machine learning algorithms and using a big dataset (around 161 MB). So, one of the input datasets household.csv could not be uploaded in Github due to size restrictions. All other documents except this dataset, including the source code are uploaded in Github. Also, all the relevant documents presentation slides, project report, source code .ipynb file including the household.csv dataset is present in google drive as well.

- Google Drive - <https://rb.gy/jdzjde>
- Github Repo – <https://github.com/darpraithatha/House-Price-Range-Prediction>

REFERENCES

- [1] Koehrsen, Will. "Beyond Accuracy: Precision and Recall." Medium, Towards Data Science, 10
- [2] Jain, Rishabh. "Decision Tree. It Begins Here." Medium, Medium, 24 June 2017
- [3] Nielsen, A., M. (1970, January 1). Neural Networks and Deep Learning
- [4] MOAWAD, A. (2019, October 8). Neural networks and backpropagation explained in a simple way
- [5] Felipe. (2017, September 1). A Quick Summary of Ensemble Learning Strategies
- [6] AdaBoost Classifier in Python. Retrieved from <https://www.datacamp.com/community/tutorials/adaboost-classifier-python>.
- [7] MLP. Retrieved from <https://www.nickgillian.com/wiki/pmwiki.php/GRT/MLP>.