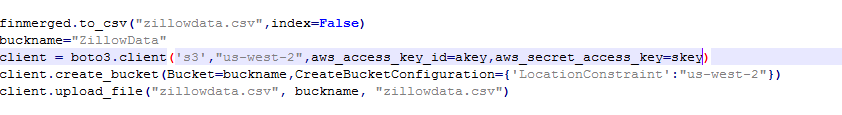
**Mid Term Assignment**

**Part 1 and Part2**

**Method to Access the cleaning Script**

1. Enter the akey and the skey in the cleaning script as below :



1. Akey=”accesskey”, SKey=”storage key”
2. Run the zillowfinalcleaning.py

We have made the script public .

**Exploratory Data Analysis**

We were provided with the Zillow Data Set that consisted of housing Data. The data consisted of properties with their features. Hence, we first went in depth to understand the various features and their significance by studing various housing sites

Understanding the Data

We first start by understanding the features and its significance in the housing domain . We understood the significance of features property land use code, property land use code, fips to name a few. Next , we identified the diffrent data types in the data to understand the features. Concepts like one hot encoding, dummy variables was understood after deep inspection of the data.

The data consisted of the log error (which is to be predicted).

Log Error=log(Zprice)-log(Zestimate)

Data Pipeline

1. Data Ingestion

2. Exploratory Data Analysis

3. Feature Selection

4. Prediction

**Data Ingestion**

We were provided with Train and Property data sets for the year 2016 and 2017. Property Dataset - The Property Data Set consists of list of features for properties Train Dataset- The Train data set consists of list of transactions that occured for the different properties. Hence, we have merged Property Data Set with Train data set based on a parcel id (which is a unique identifier). We then merged the two 2 data frames which are diffrentiated by the year column

Exploratory Data Anaysis

Methodology followed:

1. Data Type of each feature : We observed that a couple of features do not have the right data types like Tax Delinquincy Flag, Structure Tax Value Dollar Count,hashorhottub which has the data type object. These were further converted for ease of use ahead for algorithms . The values of these data types were either 'True' or 'False' or 'Yes' or 'No' These were converted to 1 or 0
2. Next we studied the correlation of Tax Data with Log error : Correlation of 1 is the Maximum correlation that can exist. We observed that these features have no strong correlation with logerror as per the heatmap. Though we do observe that structuretaxvaluedollarcnt and taxvaluedollarcnt have a correlation with logerror of 0.01 and 0.005 which is not strong but one of the highest among the the Tax Data features.

### Correlation of Numerical fields with Log Error : Next, we plotted the correlation matrix between numerical values and logerror with the help of a correlation heatmap. It is observed that the correlation is not strong but a little better than the Tax Features. These are consistently in the range 0.02.

### Correlation of Area features with Log Error : Next, we checked the correlation between area features and log error. Here, we observed that there is a relatively strong correlation between basement square feet and log error. Additionally, finishedsquarefeet12 and calculatedfinishedsquarefeet seemed to be highly correlated which we have handled under missing value handling.

### Correlation of Categorical features with Log error : Next, we tried to check correlation between categorical fields and log error. We observe that a few fields are correlated with log error but alot of the fields are negatively correlated to log error as well. Additionally categorical values like pool type id 2 and pool type id 7 so not even seem to be correlate it log error. Note :Rawcensustractandblock and Censustractandblock have very high correlation . hence, we have handled these in Handling Missing Values Section.

### 

### Count of Missing Values for each Feature :Next we plotted a Bar chart of all features wih a count of their missing values. We observe that buildingclasstypeid, basement squarefeet ,finished square feet 13 have the most number of missing values. Features CensusTrackandblock and structuretaxvaluedollarcount have the least number of missing values

### 

### Analysis of distribution of each feature

### 

### Analysis of basement square feet and log error : Since, Building Class Type Id had reasonably strong correlation with log error while analysing the correlation , we checked the bivariate distibution and univariate distribution of the relationship of these features which provided the Pearson's coefficient as well.

### 

### Analysis of Building Class Type id and Logerror : Since, Building Class Type Id had reasonably strong correlation with log error while analysing the correlation , we checked the bivariate distibution and univariate distribution of the relationship of these features which provided the Pearson's coefficient as well.

### Next we checked the number of transactions over time : The obeserved that the number of transactions were the greatest for the month of June and least for the month of December

### 

### Analysis of transaction months with log error

### Observation :From the above two graphs we see that as the number of transaction decrease the log error increases and as the number of transaction increased the mean log error decreased

### 

### Joint Plot of Month and Log error : Plotted a Joint plot of the mean log error vs Month to understand the univariate and Bivariate relations. The Bivariate distribution aids us to understand the probability distribution of the 2 random variables Log Error and Month. The Univariate axis helps understand the individual distribution of the month and the log error. Note : We do see that the plot displays a low pearson coefficient but we will explre tis furtjer with feature selection algorithms.

### 

### Next we analyzed the Count of Bedrooms and the most common Count - Next, we plotted a grah for the count of bedrooms . We observed that the most common bedrrom count in the data set was 3, followed by 2 and then 4.

### 

### Next we studied the distribution of bedroom count vs logerror in a joint plot. We observed that in the joint plot for the bed room count 2-5 was the hardest to predict.

### 

### Analysis of Bathroom Count and Log Error : The analysis of count of bathrooms indicates the most is bathroom count 2. In the joint plot below it is identified that the bathroom count 2.5 was the hardest to predict.

### 

### Similarly we identified many such features their correlation with Logerror. The Analysis is present in the Jupyter Notebook for all the features . The features analysed included Latitude, Longitude, Airconsitioning typeid, Lot size, LanTAxValuedollarCnt and CalculatedFinishedSquareFeet.

### Missing Value Analysis

### Below are the counts of missing values for each feature

### 

### 

### 

### We first dropped all records with 97% missing Data

### 

### Next, we removed records that did not have latitude mentioned in the data

### Early Prediction and Bench Mark Model

### 

### 

### Benchmark Model : best model had a MAE of 0.0613 for train and 0.718 for test data

### Further Feature selection

### We first handled missing values for poolcnt, pooltypeid2 and pooltypeid7. We observed that Pooltypeids are one hot encoded values.

### Since, we observed in the correlation matrix that finishedsquarefeet12 and calculatedfinishedsquarefeet seemed to be highly correlated, the values were all the same and hence, dropped finishedsquarefeet12.

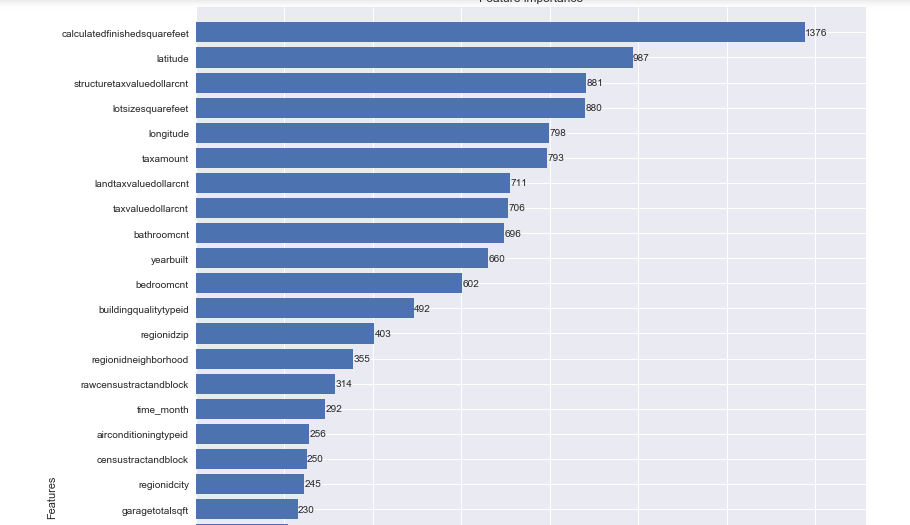
### we observed in the correlation matrix that finmerged.bathroomcnt and finmerged.calculatedbathnbr seemed to be highly correlated, the values were all the same and hence, dropped calculatedbathnbr.

### 

### Analysis of Feature selection with XGBoost

After handling missing values , we ran our data for feature selection using XGBoost. XGBoost is gradient boosted decision tree. This also helps in feature selection.

Observation : We observe that as per xgboost calculatedfinishsquarefeet,structuretxdollarvaluecnt and Latitude are the a few of the top features. Threequarterbathnbr, year and fips have beem given least importance



More Feature Selection

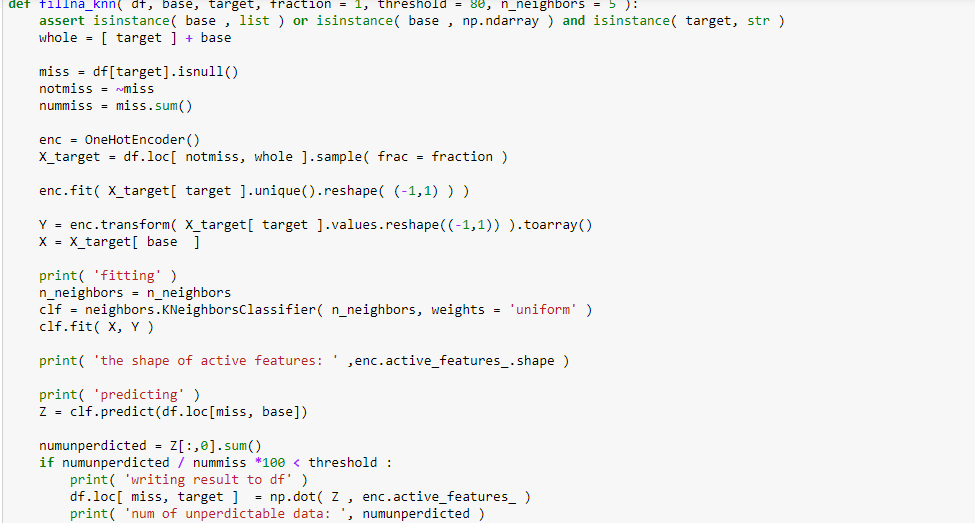
#### Relation between Fire Place Count and Fire Place Flag

In the below segment we see that there are places where fireplaceflag is true but no count is present. So we replace places where the flag is true by 1 and fill rest of fireplacecnt by 0 and drop fireplaceflag. Since, fireplace flag is not considered in the XGBoost feature selection we habe dropped the column.

1. Handled missing values in Tax Delinquicy Flag
2. we observed that some records had a garage car count but did not have Garage Total Square feet. Since, the garage square feet would vary by the garage car count , we have in accordance filled in the garage square feet values which have an existing garage car count. Those properties that had a garage car count of 0 have the garage square feet values also replaced by 0
3. correlation of finishedfloor1squarefeet and calculatedfinishedsquarefeet observe that there is a high correlation. Since, they are correlated with each other we drop finishfloor1squarefeet.
4. we observed that structuretaxvaluedollarcnt+landtaxvaluedollarcnt= taxvaluedollarcnt . hence, we have replaced the missing vaues for these respectvely by subtracting the taxvaluedollarcnt.

#### Similarly we handled missing values for rawcensustractandblock and censustractandblock, Missing Values in Tax Amount

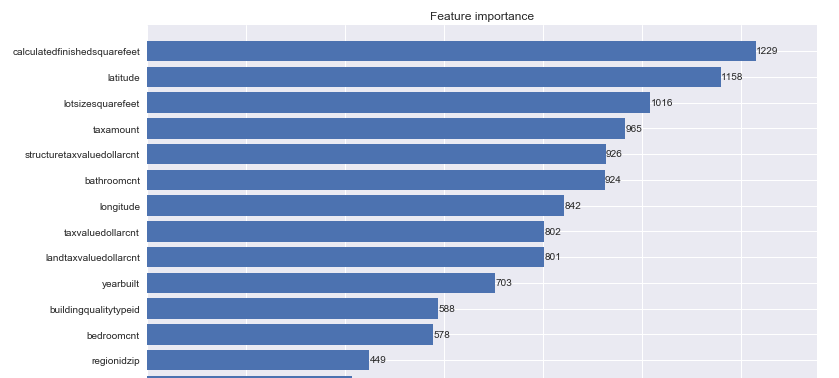
1. Handle missing values of RegionidCity, RegionidNeighborhood, Regionidzip,Unitcnt,Year built, Building quality type id, Lot size squarefeet, Building quality type id. These were handled with K Nearest Neighbors where these values were replaced by the geographically closest feature respectively.



**FEATURE SELECTION**

Feature selection was performed using Python packages like Baruta, XGBoost, Random Forest Regressor.

Feature selection by XGBoost



**Result of Feature Selection**



**Prediction Models**

The approach used for model prediction involves trying the model with different sets of features. Our trials are as follows :

Exp 1. Use of set of features (feature list present in superfinaly.csv). These features incorporate approximately all features selected by XG Boost. We have run this on Linear Regression, Random Forest and Neural Networks. We have tried to fine tune the model by the different tuning parameters.

Exp 2. Random forest has also been implemented on the clean data present in this notebook

Exp 3. We have run the Random Forest Algorithm when Categorical Values as dummy variables.

1. Categorical Values as dummy Variables (values less than 100)

2. We ran the Random Forest Algorithm with Categorical Values having values greater than 1000

The prediction models for the set 1 has been executed in a different Jupyter NoteBook namely Prediction Models as this was computationally intensive. The code is also present below and for the reader's reference we will also attach snapshots. For further details please refer the Model Prediction Notebook.

EXP 4: Ran the model without Training and Testing split

Note : The computation of each of these queries has been explained in great detail in the Jupyter Notebook EDA . Some of the prediction models have been executed in PredictionModels.ipynb.

**Inferences from the Prediction Models**

### Results of Exp 1

Studing the results of Exp 1 helped us reach the following conclusion :

1. Though the values of MAE were predicted closely and in the same range, the Random Forest Regressor gave the least MAE for Testing and Training which was 0.0687 and 0.0634 closely followed by the Linear Regression Model with 0.0688 and 0.0699 MAE. Neaural Networks also had the same MAE.
2. We observed that for this set of features the Random Forest Regressor performed the best at tuning parameters on n\_estimators of 100, min\_sample\_leaf as 20 and max\_featurs as sqrt.

The Linear Regression Model computed the fastest but with a greater MAE. The Random Forest Regressor took the longest but provided a lesser MAE and the best accuracy. Neural Networks took the long in terms of computational time and did not provide accurate results showing very few changes in the MAE inspite of tuning the parameters.

Hence, Random forest Regressor had largest computation time but provided better accuracy.

### Results of Exp 2

Running the Random Forest Regressor on features suggested by XGBoost, provided a similar accuracy as the previous case of MAE as 0.068 and 0.065 for testing and training respectively. The regressor performed its best at tuning parameters of n\_estimator as 200, max\_features as log 2 and min\_samples\_leaf as 30.

### Results of Exp 3

### Analysis of Running the Regressor with number of dummy variables

Running the Random Forest Regressor less than 100 and greater than 1000 did not show much change in accuracy. The first iteration with less dummy variables had a MAE of 0.068 and 0.067 test and train resp. For the second it was 0.0688 and 0.0686 for testing and training resp. Both performed at the opimum of 200 estmators and log 2 max\_features min\_leaf\_size20 and 25 respectively.

### Results of Exp 4

### Analysis of Running the Regressor with Train and Test Split different

### Since, we realized that as the competition of the testing data was the data after the 10th month of the data. We tried to test the data by removing the Testing and Training split.

### We achieved our best results as follows:

### 

### 

**Our optimum result was obtained at a MAE of Testing of 0.065534**

**Our optimum result was obtained at a MAE of Testing of 0.06456**

Please access this link to see the cleaned .csv file which came in Amazon S3 Bucket.

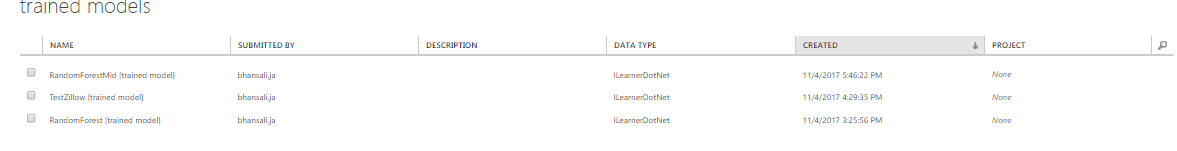
<https://s3-us-west-2.amazonaws.com/zillodata/zillowdata.csv>

**Part 3**

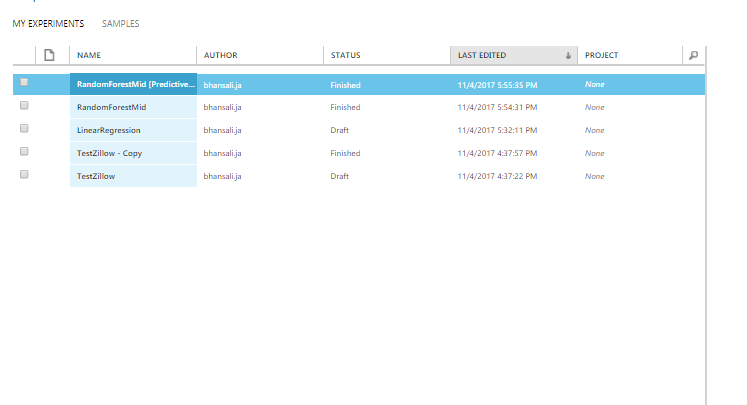
**Deploying using Azure ML**

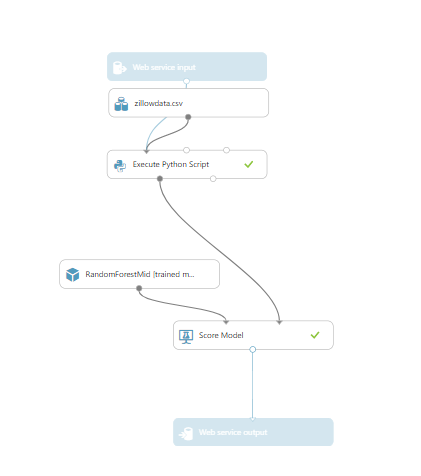
**Steps followed:**

1. Created an azure machine learning workspace
2. Created an experiment
3. Added the cleaned data set and selected visualize
4. We ran a python script and dropped the columns parcel id and transactiondate.
5. After that we split the data, this is done before we can train the model by using Split Data Module.
6. We now pick the model to be trained which was our random forest regressor as it was our best model in Step 2 and connected the train model module to Decision Forest Regression Module.
7. While the train model module is selected, click the Launch Column selector button.
8. Select the logerror column, because it is the column we want the Model to Predict.
9. Click Run to populate the data to the Train Model module.



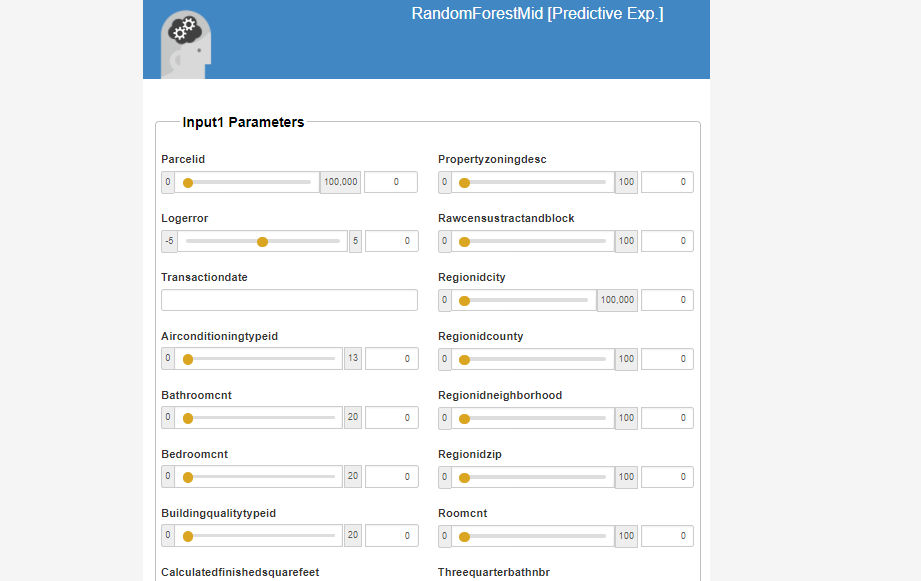
1. We now need to score the Model against the data held back with the Split Data Module.
2. Drag and Drop a Score Model module on the design surface, and connect it to the Train Model module and the Split Data model.
3. Click Run to populate to the Score Model module.
4. Visualize the Score Model results.
5. We now need to evaluate the performance of the Model, Drag and Drop an Evaluate Model module on the design surface, and connect it to Score Model module.
6. Click Run to populate the data Evaluate Model module.
7. Now, that we have a Model, we need to operationalize it, click Run to run the entire experiment.
8. Click the Set Up Web Service button and select Predictive Web Service and now we have web service that can be consumed programmatically.
9. Now, click on the Configuration button, click the world icon to navigate to the menu that will show our web service, now click on the web service we were working on to return to its configuration.





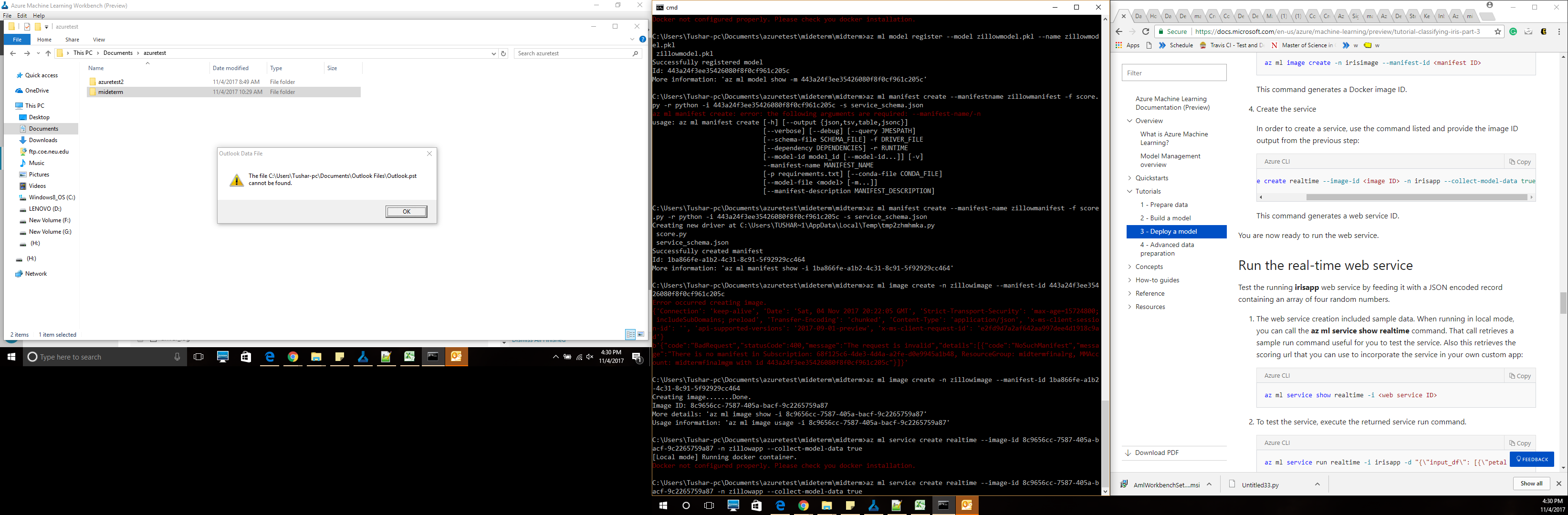
1. Click on the Request/Response row and we can copy the Post url and update and configure the REST API created on workbench and update the settings through deploy.

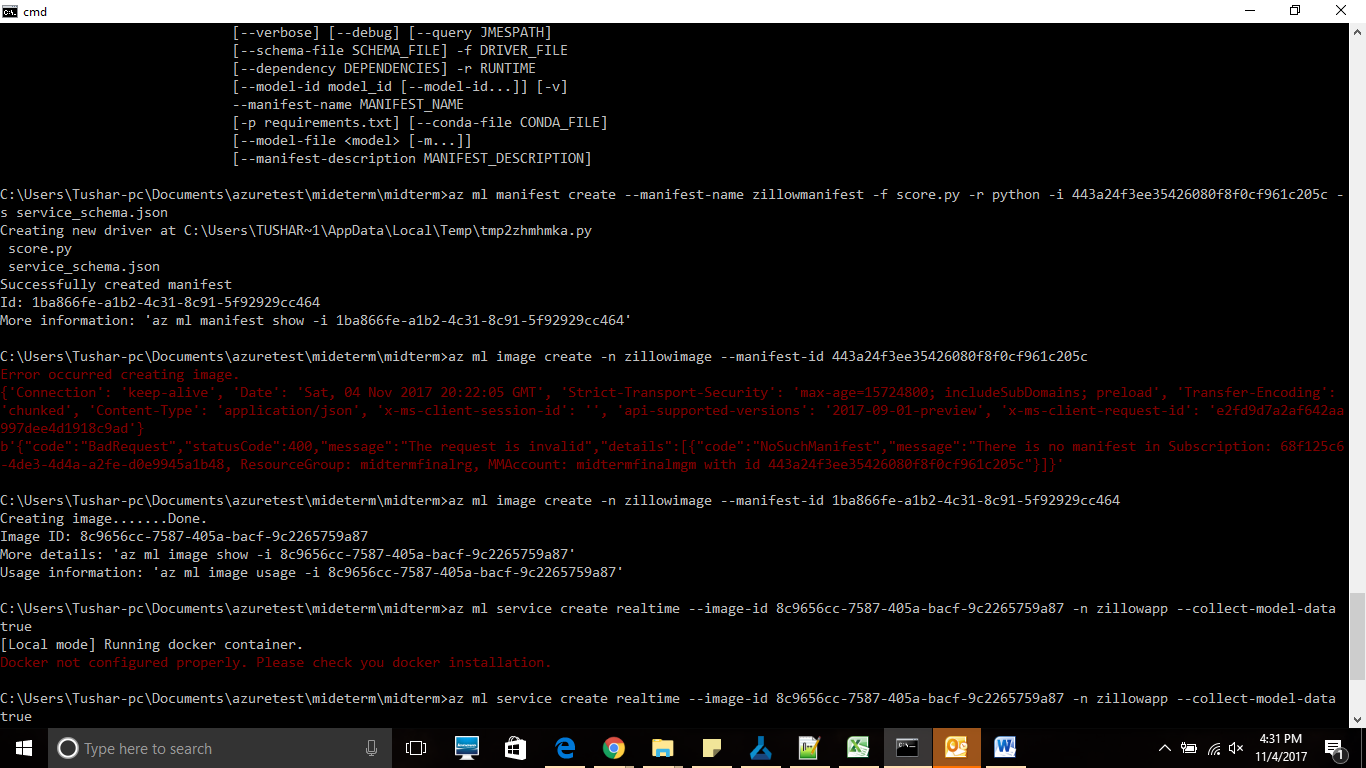
Link to our REST API : <http://zillowdataapi.azurewebsites.net/>

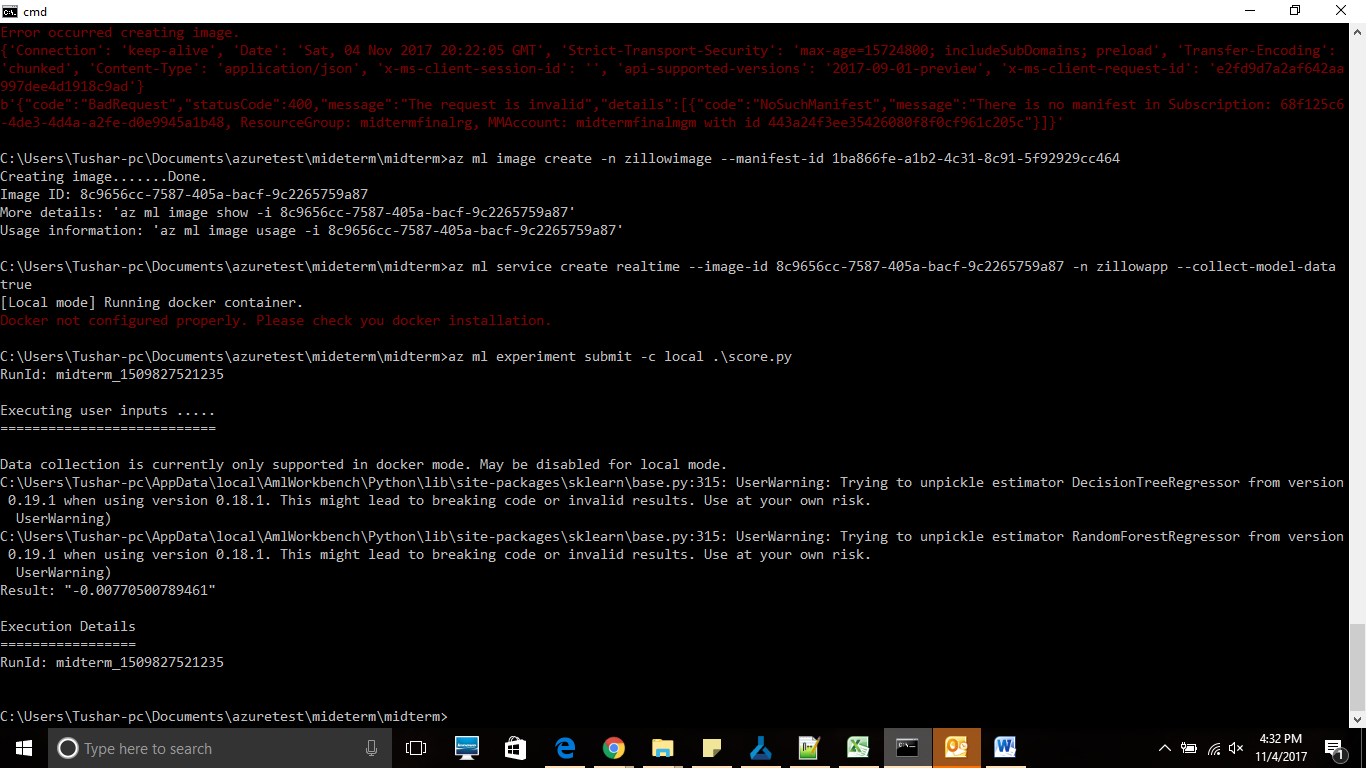


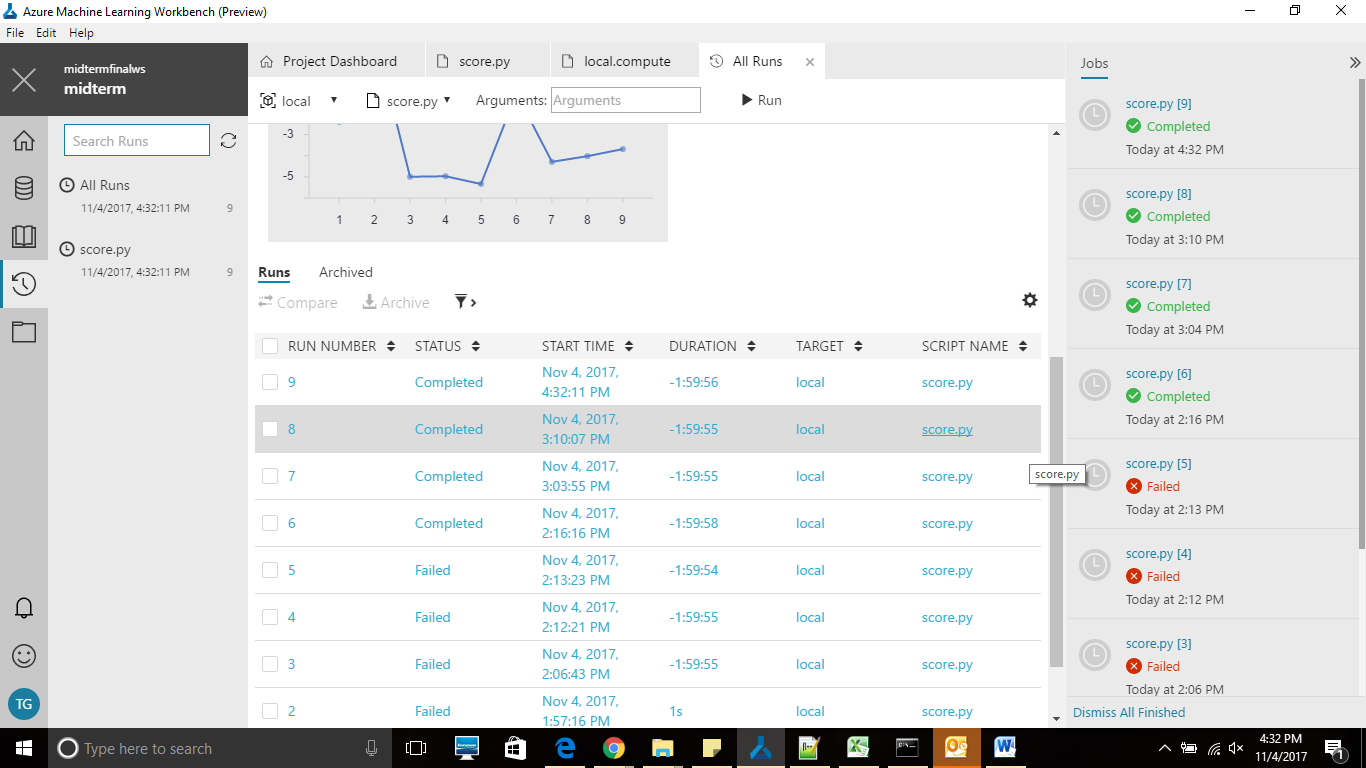
Attempted Azure ML CLI

We have used azure ml cli. The CLI is used to deploy a web service. Azure ML helps model to be deployed. This model is made an image in docker . Unfortunately the docker configuration does not work on Windows. We have implemented the steps till creating the image and receiving the app id. Additionally, the refrences indicate a local deployment , hence we have implemented this and also created our own web app that handles the model deployment. Please find attached the screenshots for the same.



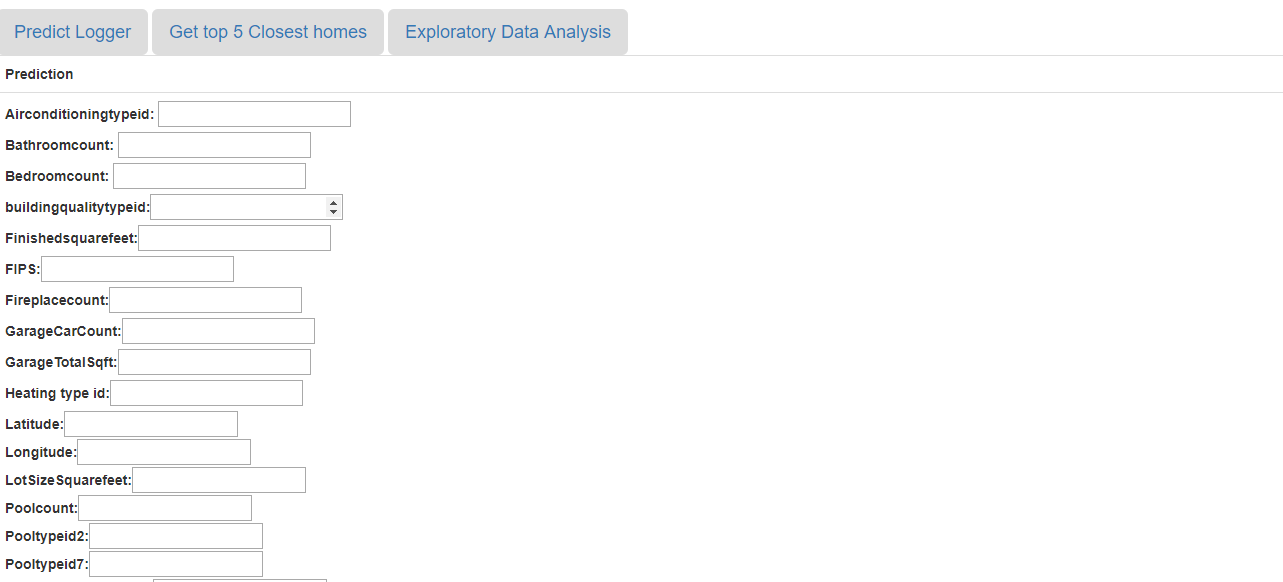






We have created a web app that deploys our model. The webapp is created in python and is using the ec2 instance.

**Web Deployment using a WebAPP**



Part 4

We have used the Harvesian Stored procedure to get the 10 closest homes when a user enters a latitude and longitude. We have created an HTML page called Index.html that takes the latitude and longitude and in the backend hits the mysql to generate the 10 closest homes. We have used a python flask to implement this.

