ANALYSIS ON <u>LOAN LENDING CLUB</u> USING

Report By:
Ankit Bhayani
Rajat Agrawal
Vishakha Sawant

OBJECTIVE

In Assignment Part 1, we analyze the Lending Loan Club Loan Approval data and Loan Reject data. In this part, we are going to build and evaluate the model by applying different Classification and Regression Machine Learning Algorithms. We have divided our problem statement into 2 section:

- Determine whether to Approve Loan or Not
- If Approved, Predict the interest rate

INTRODUCTION OF AZURE ML

Azure Machine Learning was designed for applied machine learning. We can use algorithms in simple drag-and-drop interface and go from idea to deployment in a matter of clicks.

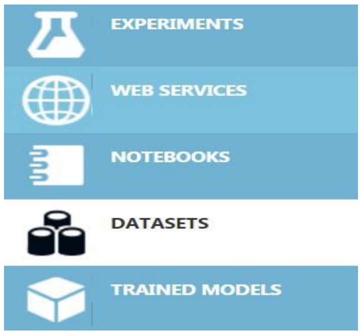
We can deploy our model into production as a web service, ml gives us a web service that can be called from any device, anywhere and that can use any data source, using REST service.

DESIGN AND IMPLEMENTATION

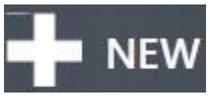
We should import our dataset to Azure ML as follows:

IMPORTING DATASET

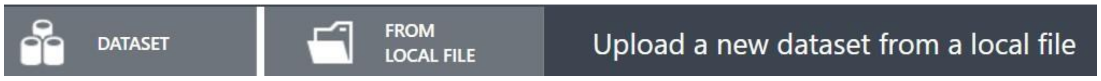
1. Click 'Datasets' on the left menu of the studio.



2. Now click on the '+' to import the dataset from the local storage.



3. When NEW is click it will give following option:



4. choose your dataset and follow the instructions to finish import.

CLASSIFICATION

When the data are being used to predict a category, supervised learning is also called classification. This is the case when assigning an image as a picture of either a 'cat' or a 'dog'. When there are only two choices, this is called **two-class** or **binomial classification**.

In case of Loan Lending Club, Loan Request is classified as "Approved" and "Reject".

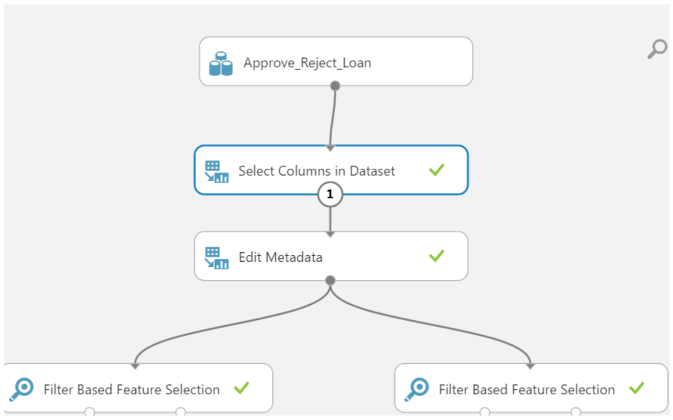
COMMON STEPS FOR ALL CLASSIFICATION

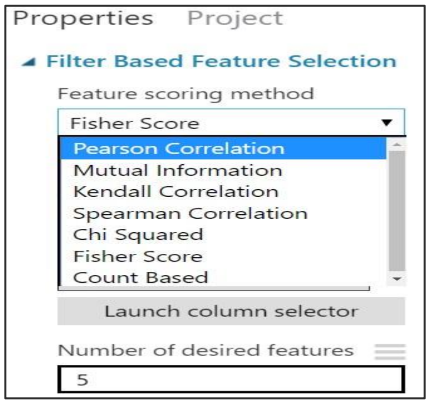
Feature selection, Normalization, Split data, Train Model, Score Model, Evaluate Model are steps in all the models. We will go through the steps as follows.

FEATURE SELECTION EXPERIEMNT

Before we proceed with our experiment, we have done feature selection in a separate experiment.

Since we have 10 different features which consist of 5 Main features and 3 derived feature, we apply the feature selection technique to come up with the best feature used to classify the applicant Loan request.


Approve_Reject_Loan
Select Columns in Dataset
1
Edit Metadata
Filter Based Feature Selection
Filter Based Feature Selection



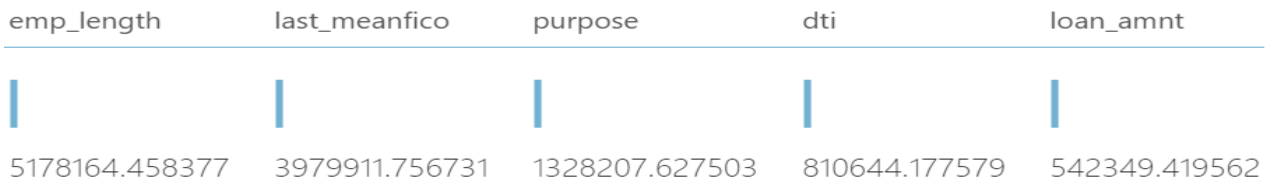
Filter Based Feature Selection

Feature scoring method: Fisher Score

Number of desired features: 5

We used 2 different method to find the best feature for our model:

- Pearson Correlation
- Chi-Squared



emp_length	last_meanfico	purpose	dti	loan_amnt
5178164.458377	3979911.756731	1328207.627503	810644.177579	542349.419562

So, we shortlist the above features to use it in our clustering experiment.

DRAG DATASET TO CLASSFICATION EXPERIMENT

We initially start with Jupyter notebook to see the accuracy of the various model required for the assignment.

```
#Keep the following 10 features (variables) which are important
cols_to_keep = ['loan_amnt', 'last_meanfico', 'dti','addr_state','emp_length','purpose']
train, test = sklearn.cross_validation.train_test_split(df_data_1, train_size = 0.7)
train_y = train['status']
test_y = test['status']
train_x = train[cols_to_keep]
test_x = test[cols_to_keep]
```

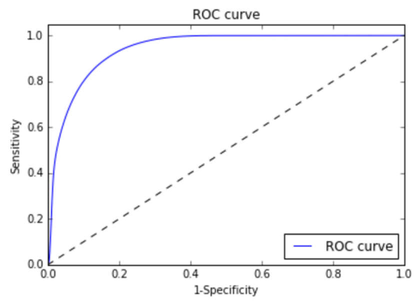
```
from sklearn import preprocessing
# Discreet value integer encoder
label_encoder = preprocessing.LabelEncoder()
# State is string and we want discre integer values
train_x['state'] = label_encoder.fit_transform(train_x['addr_state'])
test_x['state'] = label_encoder.fit_transform(test_x['addr_state'])
train_x['purp'] = label_encoder.fit_transform(train_x['purpose'])
test_x['purp'] = label_encoder.fit_transform(test_x['purpose'])
```

```
train_x = train_x._get_numeric_data()
test_x = test_x._get_numeric_data()
```

Logistic Regression:

```
#Logistic regression
log_reg = LogisticRegression()
log_reg.fit(train_x, train_y)
print ("Intercept is ",log_reg.intercept_)
print("Coefficient is ",log_reg.coef_)
y_pred=log_reg.predict(test_x)

#calculate ROC curve
preds = log_reg.predict_proba(test_x)[:,1]
calculate_roc_curve(test_y, preds,2)

#calculate Confusion Matrix
calculate_confusion_matrix(test_y, y_pred)

print(accuracy_score(test_y, y_pred))
```

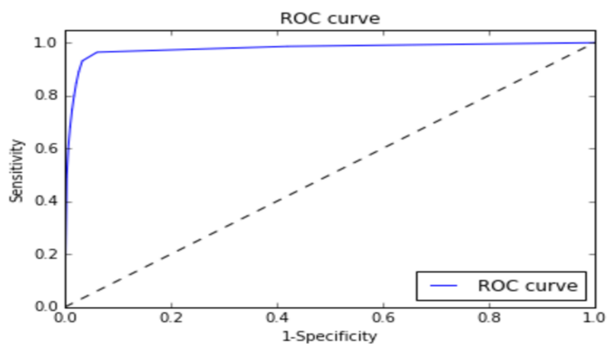
```
('Intercept is ', array([-4.35235659]))
('Coefficient is ', array([[ -2.07088278e-05,   6.27878832e-03,  -2.32898171e-02,
          3.20780048e-01,  -1.67046429e-02,  -2.21743219e-02]]))
```



```
[[3256338   67854]
 [ 213655  182523]]
0.924333063647
```

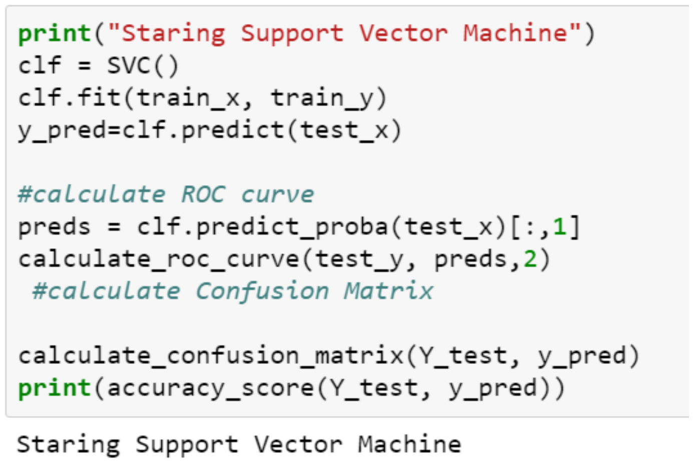
Random Dense Forest:

```
#Random Forest
rf = RandomForestClassifier(n_jobs=2)
rf.fit(train_x, train_y)
preds = rf.predict_proba(test_x)[:,1]
y_pred=rf.predict(test_x)
#calculate ROC curve
calculate_roc_curve(test_y, y_pred,2)
 #calculate Confusion Matrix

calculate_confusion_matrix(test_y, y_pred)
print(accuracy_score(test_y, y_pred))
```



```
[[3266240   57952]
 [  79249  316929]]
0.963121678758
```

SVM (Scalar Vector Machine):



```
print("Staring Support Vector Machine")
clf = SVC()
clf.fit(train_x, train_y)
y_pred=clf.predict(test_x)

#calculate ROC curve
preds = clf.predict_proba(test_x)[:,1]
calculate_roc_curve(test_y, preds,2)
 #calculate Confusion Matrix

calculate_confusion_matrix(Y_test, y_pred)
print(accuracy_score(Y_test, y_pred))
```

```
Staring Support Vector Machine
```

After performing our analysis on Jupyter notebook, we move to Azure ML platform to run SVM and Neural network following Logistic classification and Dense Random forest.

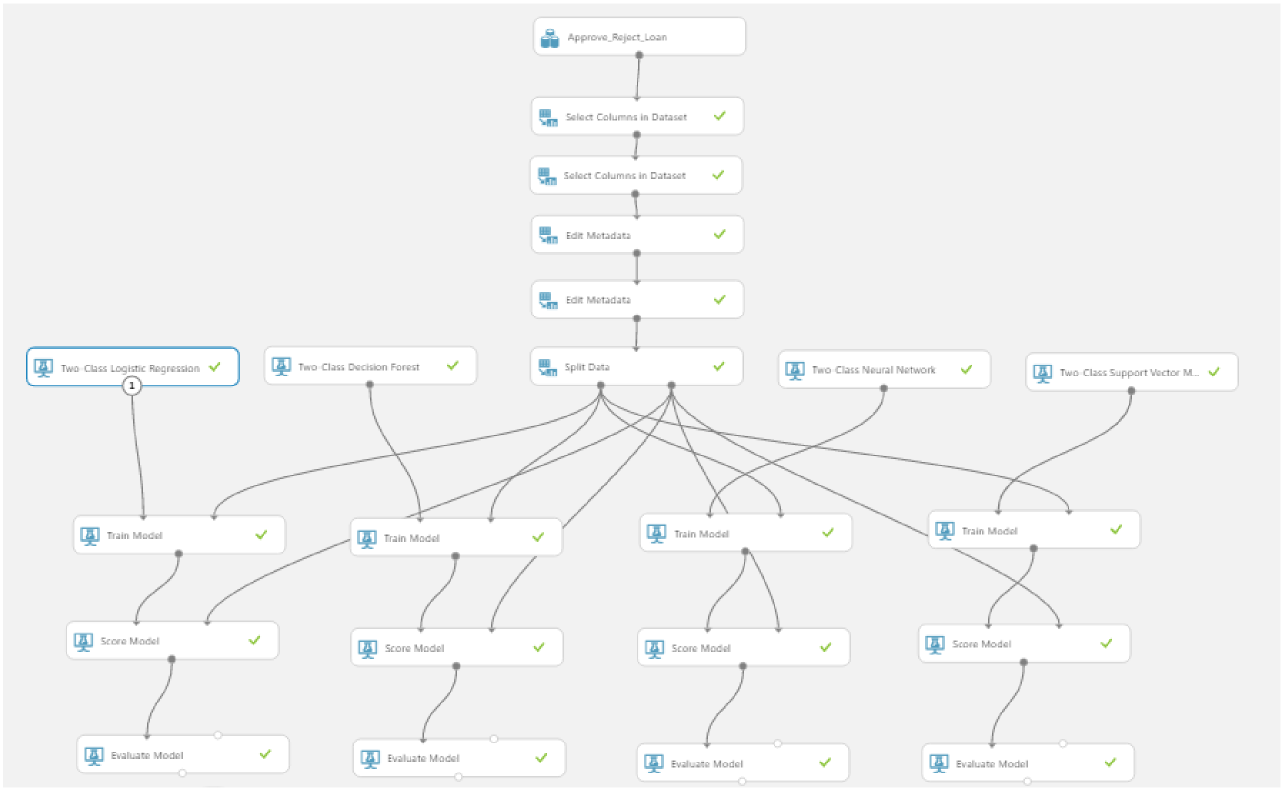
SELECT COLUMNS IN DATASET

For Training our model in Azure ML, we will select only those columns required for the modeling as we decided by Feature Selection experiment shown above and preprocess the column by Numerical and categorical fields.

SPLIT DATA: TESTING AND VALIDATION

A common strategy is to take all available labeled data, and split it into training and evaluation subsets, usually with a ratio of 70-80 percent for training and 20-30 percent for evaluation. The ML system uses the training data to train models to see patterns, and uses the evaluation data to evaluate the predictive quality of the trained model. We use 70% - 30% for training and testing.

IMPLEMENTING CLASSIFICATION MODEL ON AZURE ML



We implemented the following four classification algorithm for classifying user request:

- Logistic Regression
- Random Dense Forest
- Neural Network
- SVM

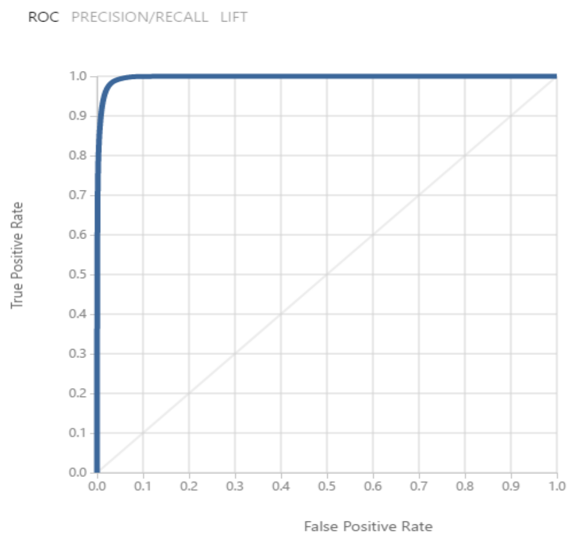
Two Class Decision Forest

We will train the Random Forest model given by Azure ML.

The 70% of the split data is feed into Train Model component along with Random Forest component.

The ROC curve and accuracy is good with accuracy of 98%

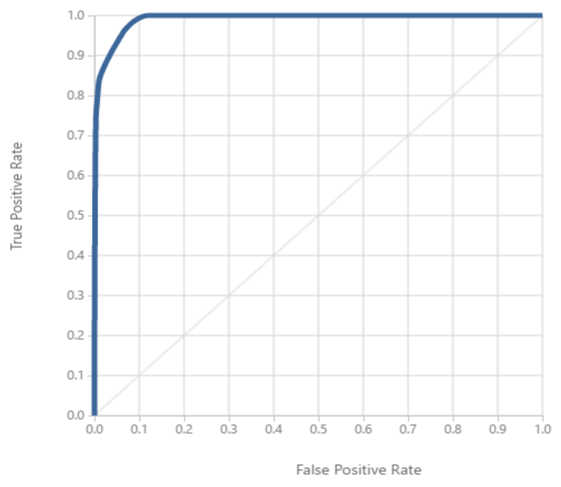
The Accuracy is good so we can finalize the settings of this trained model, otherwise we could have tuned the parameters of the algorithm component.



True Positive	False Negative	Accuracy	Precision
358318	38879	0.982	0.926
False Positive	**True Negative**	**Recall**	**F1 Score**
28807	3294366	0.902	0.914

NEURAL NETWORK

The ROC curve is shown is visualized as below, having a good accuracy of 97%.



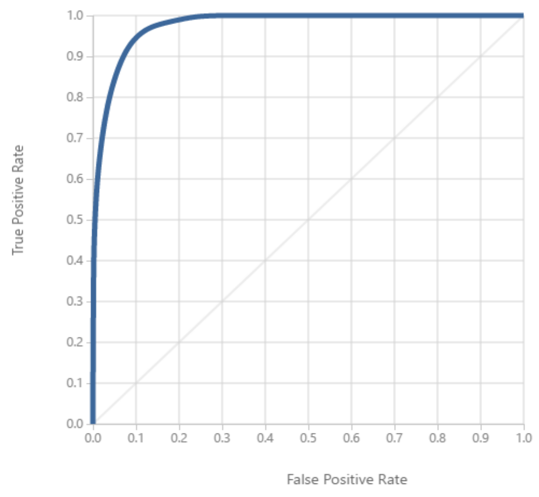
True Positive	False Negative	Accuracy	Precision
306973	90224	0.971	0.950
False Positive	**True Negative**	**Recall**	**F1 Score**
16295	3306867	0.773	0.852

LOGISTIC REGRESSION

We will be using the Logistic Regression component given by Azure ML studio.

The 70% of the split data is feed into Train Model component along with Random Forest component.

The visualization of the Evaluate Model is shown below:

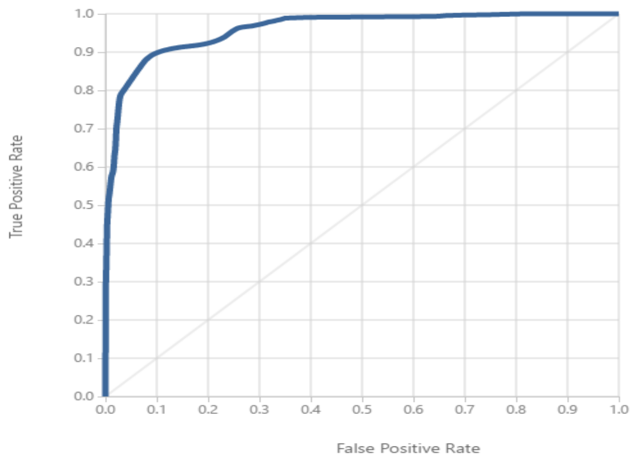


True Positive	False Negative	Accuracy	Precision
262769	134428	0.949	0.825
False Positive	**True Negative**	**Recall**	**F1 Score**
55759	3267403	0.662	0.734

Scalar Vector Machine

You can use the **Two-Class SVM** module to create a trained and test the model.

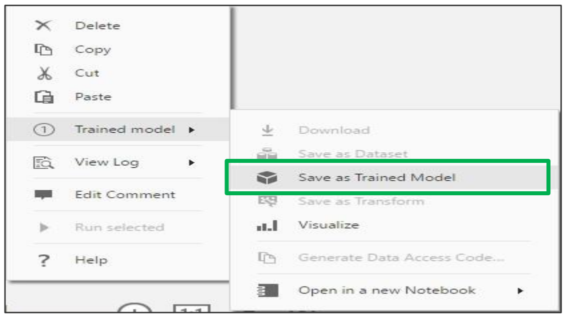
The ROC curve is shown is visualized as below, having a good accuracy of 97%.



True Positive	False Negative	Accuracy	Precision
240413	156784	0.943	0.815
False Positive	**True Negative**	**Recall**	**F1 Score**
54643	3268519	0.605	0.695

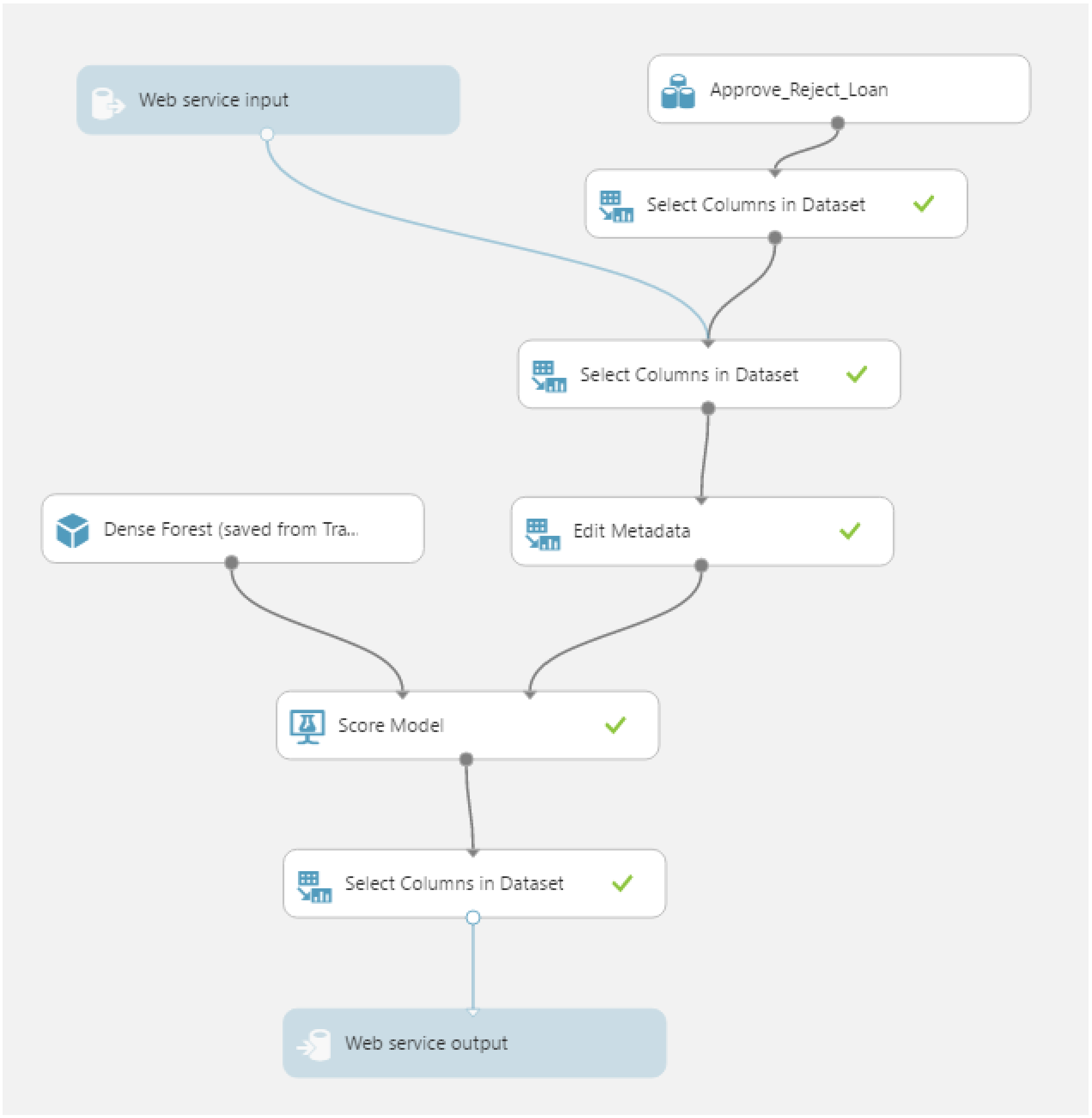
Saving TRAINED MODEL

After we run all the models and is satisfied with the Scored Model output, we proceed to save the model by right clicking it. These saved will be used when we will setup our web service.

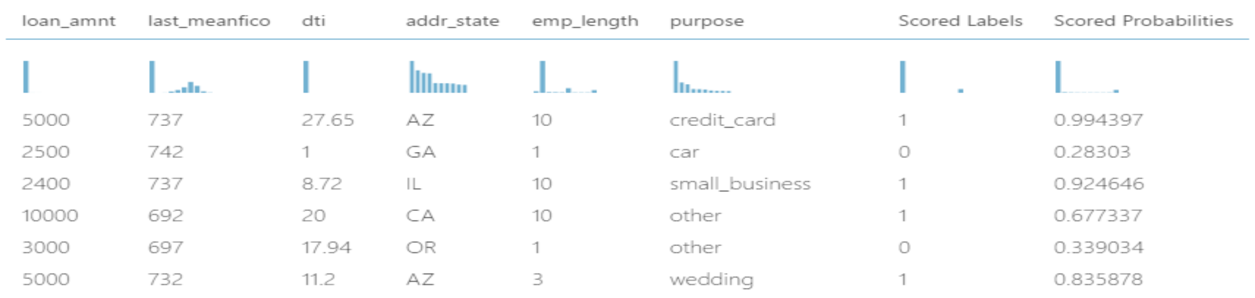


SETUP WEB SERVICE: CLASSIFICATION

Our model upon setting up as web service looks like below, in this we have dragged our saved models.


Web service input
Approve_Reject_Loan
Select Columns in Dataset
Select Columns in Dataset
Dense Forest (saved from Tra...
Edit Metadata
Score Model
Select Columns in Dataset
Web service output

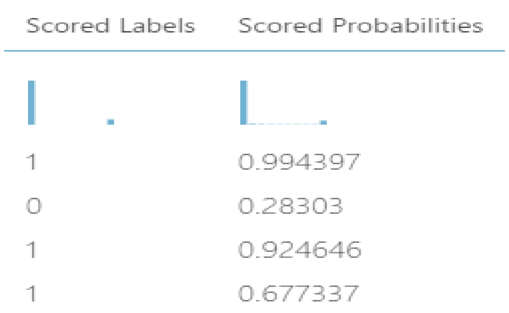
When we visualize the Score Model component we get scored labels and scored probabilities along with other features.



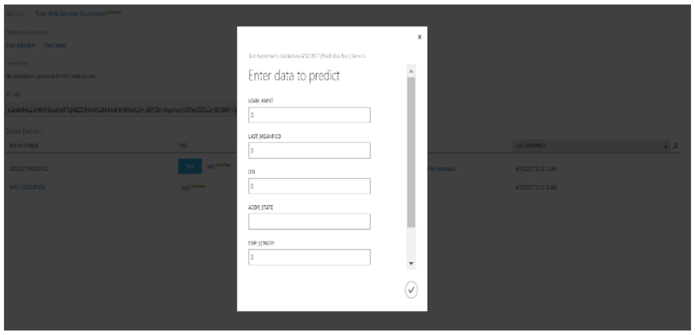
loan_amnt	last_meanfico	dti	addr_state	emp_length	purpose	Scored Labels	Scored Probabilities
5000	737	27.65	AZ	10	credit_card	1	0.994397
2500	742	1	GA	1	car	0	0.28303
2400	737	8.72	IL	10	small_business	1	0.924646
10000	692	20	CA	10	other	1	0.677337
3000	697	17.94	OR	1	other	0	0.339034
5000	732	11.2	AZ	3	wedding	1	0.835878

Scored Probabilities: We will consider taking this value as an output of web service to allow user to choose a model based on accuracy instead of using his own judgment.

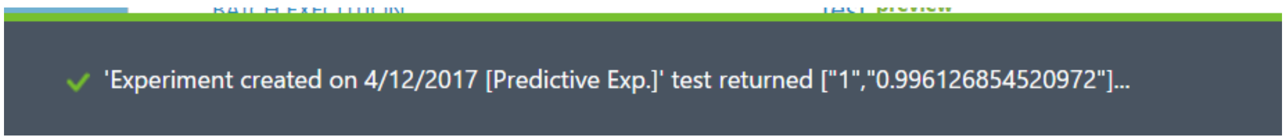
We are using 'Select Column in Dataset' component to extract just these 2 columns.



Scored Labels	Scored Probabilities
1	0.994397
0	0.28303
1	0.924646
1	0.677337



The output of this experiment is JSON format :



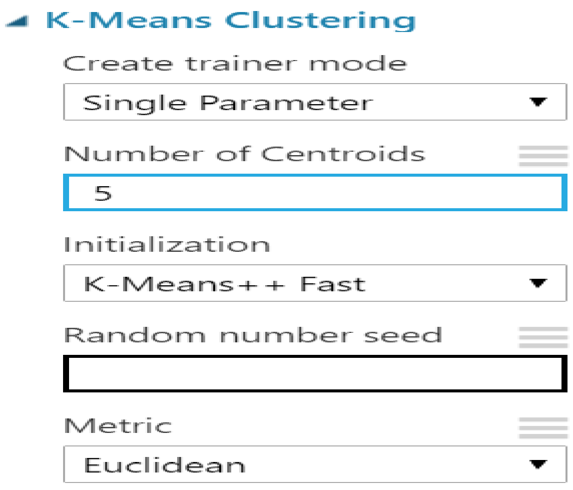
'Experiment created on 4/12/2017 [Predictive Exp.]' test returned ["1","0.996126854520972"]...

CLUSTERING

Cluster analysis classifies a set of observations into two or more mutually exclusive unknown groups based on combinations of variables. The purpose of cluster analysis is to discover a system of organizing observations, their characteristics, into groups, where members of the groups share properties in common.

K-MEANS CLUSTERING

We can use the **K-Means Clustering** module to create an untrained K-means clustering model. K- means is one of the simplest and the best known *unsupervised* learning algorithms, and can be used for a variety of machine learning tasks, such as detecting abnormal data, clustering of text documents, and analysis of a dataset prior to using other classification or regression methods.



K-Means Clustering

Create trainer mode: Single Parameter

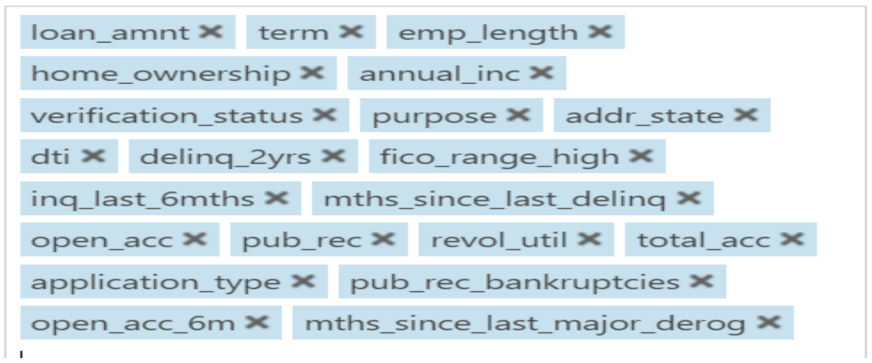
Number of Centroids: 5

Initialization: K-Means++ Fast

Random number seed:

Metric: Euclidean

We have selected columns as below for clustering


loan_amnt ✕ term ✕ emp_length ✕
home_ownership ✕ annual_inc ✕
verification_status ✕ purpose ✕ addr_state ✕
dti ✕ delinq_2yrs ✕ fico_range_high ✕
inq_last_6mths ✕ mths_since_last_delinq ✕
open_acc ✕ pub_rec ✕ revol_util ✕ total_acc ✕
application_type ✕ pub_rec_bankruptcies ✕
open_acc_6m ✕ mths_since_last_major_derog ✕

Splitting the data follow what we did in previous experiments 70% 30% split for test and train.

Finally, after the training is done we have assigned below columns to assign to cluster.

000,000
000,000
000,000
0
000,000
000,000
000,000
000,000
Principal Component 2
Cluster 0
Cluster 4
Cluster 2
Cluster 3
Cluster 1

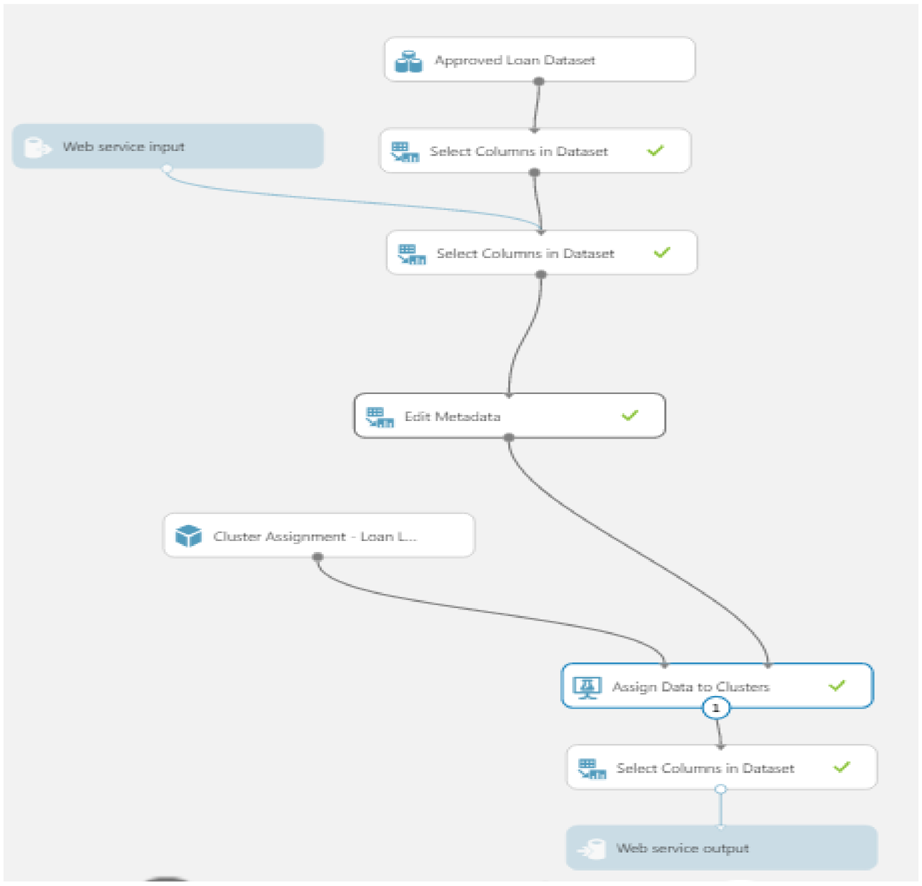
MANUAL CLUSTERING

For Manual clustering, we choose Term and Fico Score as our main factor to cluster the user and predict their interest rate.

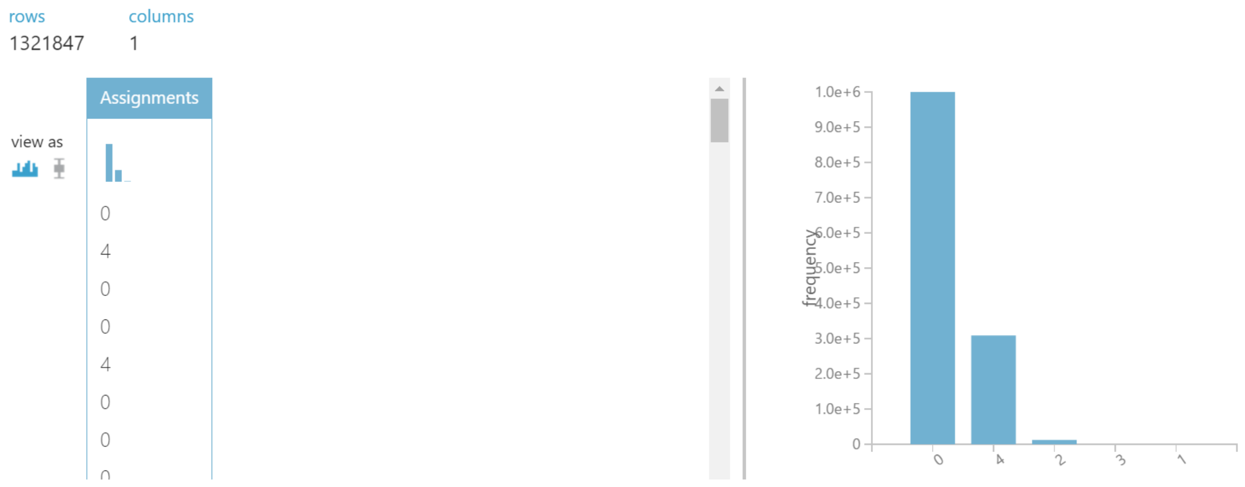
We first divide the user based on the Fico score bins and then split them on the Term (36 or 60 months) and created 6 csv file to train our model and predict our interest rate.

SETUP WEB SERVICE: CLUSTERING

Our model upon setting up as web service looks like below, in this we have dragged our saved models.

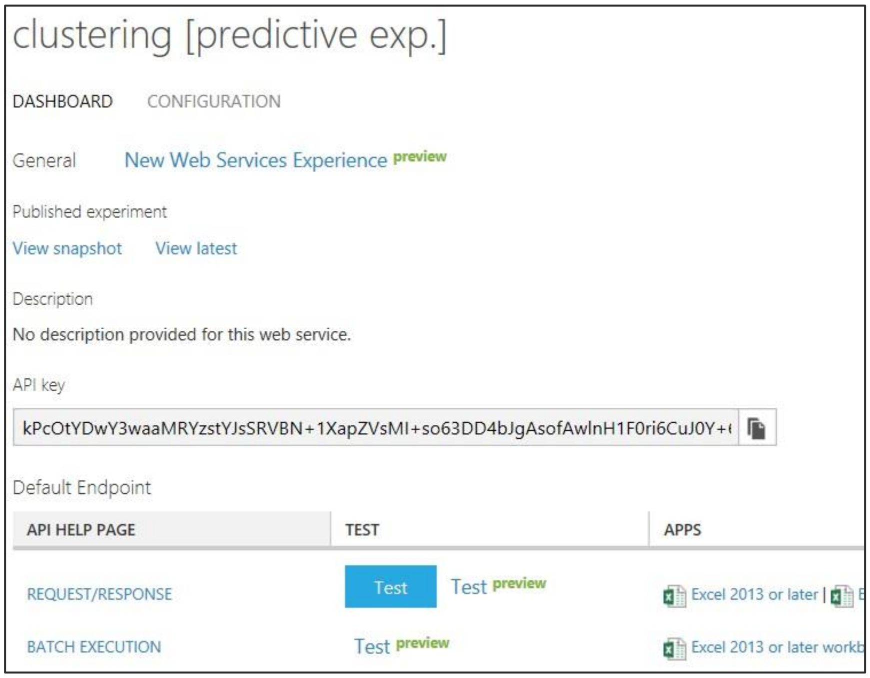

Approved Loan Dataset
Web service input
Select Columns in Dataset
Select Columns in Dataset
Edit Metadata
Cluster Assignment - Loan L...
Assign Data to Clusters
1
Select Columns in Dataset
Web service output

The Output is shown as:



DEPLOY THE WEB SERVICE

Like what we did for classification we can Test out web service.



clustering [predictive exp.]

DASHBOARD CONFIGURATION

General New Web Services Experience preview

Published experiment

View snapshot View latest

Description

No description provided for this web service.

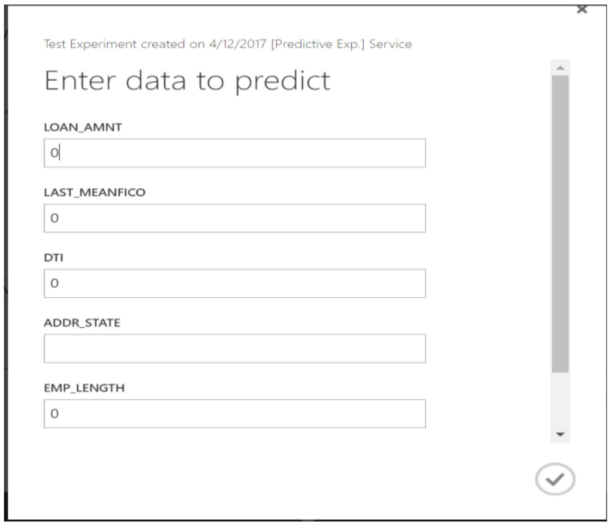
API key

kPcOtYDwY3waaMRYzstYJsSRVBN+1XapZVsMI+so63DD4bJgAsofAwlnH1F0ri6CuJ0Y+

Default Endpoint

API HELP PAGE	TEST	APPS
REQUEST/RESPONSE	Test Test preview	Excel 2013 or later |
BATCH EXECUTION	Test preview	Excel 2013 or later workb

We will fill out the data to the experiment

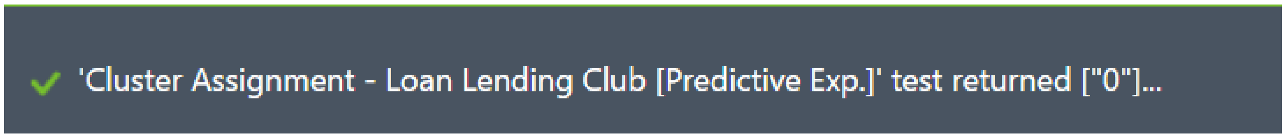


Test Experiment created on 4/12/2017 [Predictive Exp.] Service

Enter data to predict

LOAN_AMNT	0
LAST_MEANFICO	0
DTI	0
ADDR_STATE	
EMP_LENGTH	0

The output of this experiment is JSON format, which mentions different clusters.



'Cluster Assignment - Loan Lending Club [Predictive Exp.]' test returned ["0"]...

REGRESSION

Regression is most commonly used algorithm when there is a need a for Predicting a value. Regression tries to predict a real valued output (numerical value) of some variable for that individual. An example regression problem would be: "What will be the cost of a given house?" . The variable to be predicted here is housing price, and a model could be produced by looking at other, similar houses in the population and their historical prices.

COMMON STEPS FOR ALL REGRESSION

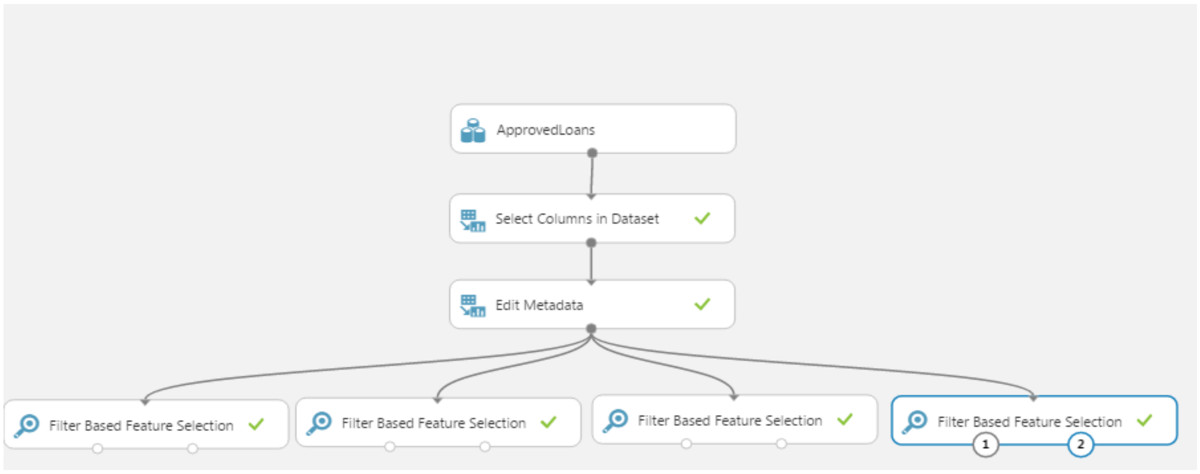
Initial Feature Selection, Categorical Casting, split data, Train Model, Score Model, Evaluate Model are steps in all the models. We will go through the steps as follows.

INITIAL FEATURE SELECTION EXPERIEMNT

Since we have 105 features in our cleaned file, not every feature in its current form is expected to contain predictive value to the model, and may mislead or add noise to the model. Some low quality features were removed to improve the model's performance. Low quality includes lack of representative categories, too many missing values, or noisy features.

1. We start the experiment by dragging the dataset.
2. Begin by identifying columns that add little-to-no value for predictive modeling. These columns will be dropped. The columns which will be input to the model.

   We select the column names from the original dataset by using various feature selection technique:


ApprovedLoans
Select Columns in Dataset
Edit Metadata
Filter Based Feature Selection
Filter Based Feature Selection
Filter Based Feature Selection
Filter Based Feature Selection
1
2

   Based on the four-feature selection algorithm we came up with 21 columns with major significance on predicting interest rate:

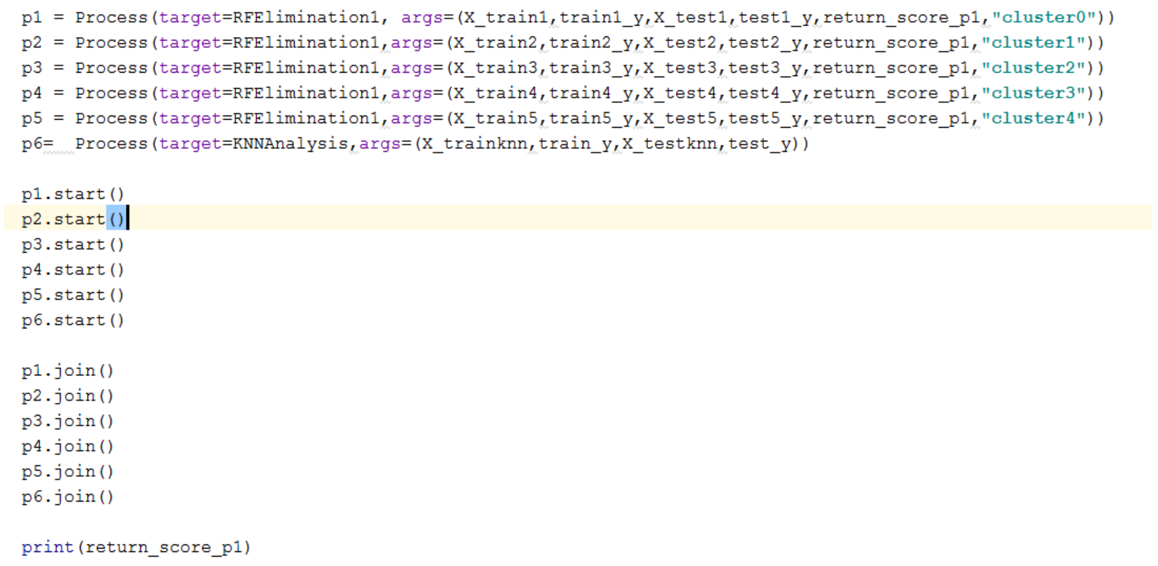
   'term','purpose','dti','loan_amnt','annual_inc','home_ownership','addr_state','fico_range_high','emp_length','application_type','verification_status','revol_util','inq_last_6mths','open_acc_6m','pub_rec','pub_rec_bankruptcies','delinq_2yrs','open_acc','total_acc','mths_since_last_delinq','mths_since_last_major_derog'

3. Categorical casting

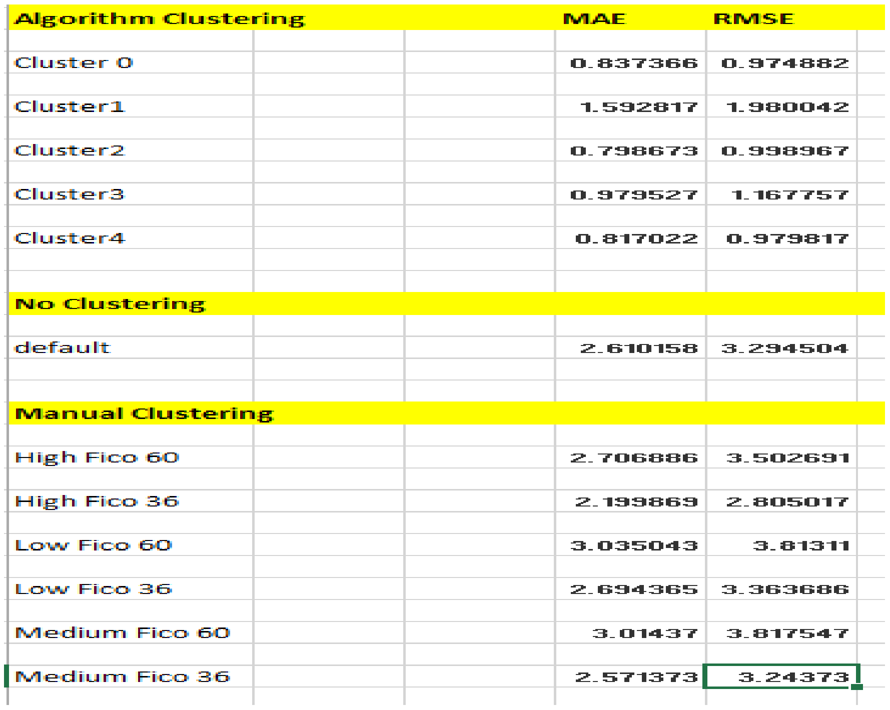
   Nominal categorical features were identified and cast to categorical data types using the meta data editor to ensure proper mathematical treatment by the machine learning algorithm. To cast these columns, drag in the metadata editor. Specify the columns to be cast, then change the "Categorical" parameter to "Make categorical".

With this, we also start working on the Jupiter notebook to get the best variables present in the cluster.

Performed Variable selection using RFEElimination per segment/cluster. We executed parallel processing for calculating score per cluster.



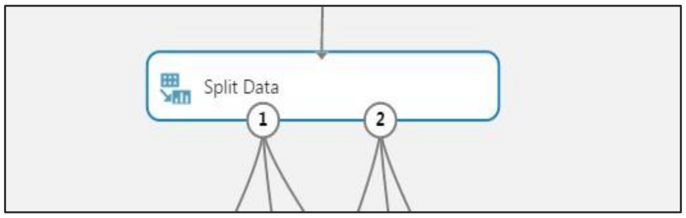
```
p1 = Process(target=RFElimination1, args=(X_train1,train1_y,X_test1,test1_y,return_score_p1,"cluster0"))
p2 = Process(target=RFElimination1,args=(X_train2,train2_y,X_test2,test2_y,return_score_p1,"cluster1"))
p3 = Process(target=RFElimination1,args=(X_train3,train3_y,X_test3,test3_y,return_score_p1,"cluster2"))
p4 = Process(target=RFElimination1,args=(X_train4,train4_y,X_test4,test4_y,return_score_p1,"cluster3"))
p5 = Process(target=RFElimination1,args=(X_train5,train5_y,X_test5,test5_y,return_score_p1,"cluster4"))
p6=  Process(target=KNNAnalysis,args=(X_trainknn,train_y,X_testknn,test_y))

p1.start()
p2.start()
p3.start()
p4.start()
p5.start()
p6.start()

p1.join()
p2.join()
p3.join()
p4.join()
p5.join()
p6.join()

print(return_score_p1)
```

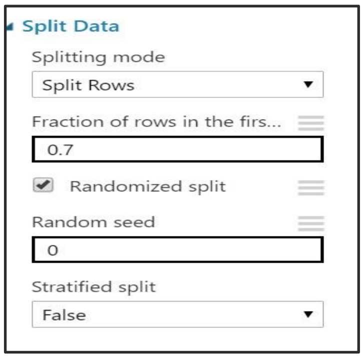


Algorithm Clustering	MAE	RMSE
Cluster 0	0.837366	0.974882
Cluster1	1.592817	1.980042
Cluster2	0.798673	0.998967
Cluster3	0.979527	1.167757
Cluster4	0.817022	0.979817
No Clustering		
default	2.610158	3.294504
Manual Clustering		
High Fico 60	2.706886	3.502691
High Fico 36	2.199869	2.805017
Low Fico 60	3.035043	3.81311
Low Fico 36	2.694365	3.363686
Medium Fico 60	3.01437	3.817547
Medium Fico 36	2.571373	3.24373

SPLIT THE DATA

It is extremely important to randomly partition your data prior to training an algorithm to test the validity and performance of your model. A predictive model is worthless to us if it can only accurately predict known values. Withhold data represents data that the model never saw when it was training its algorithm. This will allow you to score the performance of your model later to evaluate how well the model can predict future or unknown values.Randomly split and partition the data into 70% training and 30% scoring using the split module. Drag in a "Split" module. It is usually industry practice to set a 70/30.





Split Data

Splitting mode: Split Rows

Fraction of rows in the firs...: 0.7

✓ Randomized split

Random seed: 0

Stratified split: False

Now that the data is ready, constructing a predictive model consists of training and testing. We'll use our data to train the model, and then we'll test the model to see how closely it's able to predict.

LINEAR REGRESSION

Regression is a machine learning used to predict a numeric outcome. Linear regression attempts to establish a linear relationship between independent variables and an outcome variable, or *dependent variable*, which is also numeric.

NEURAL NETWORK REGRESSION

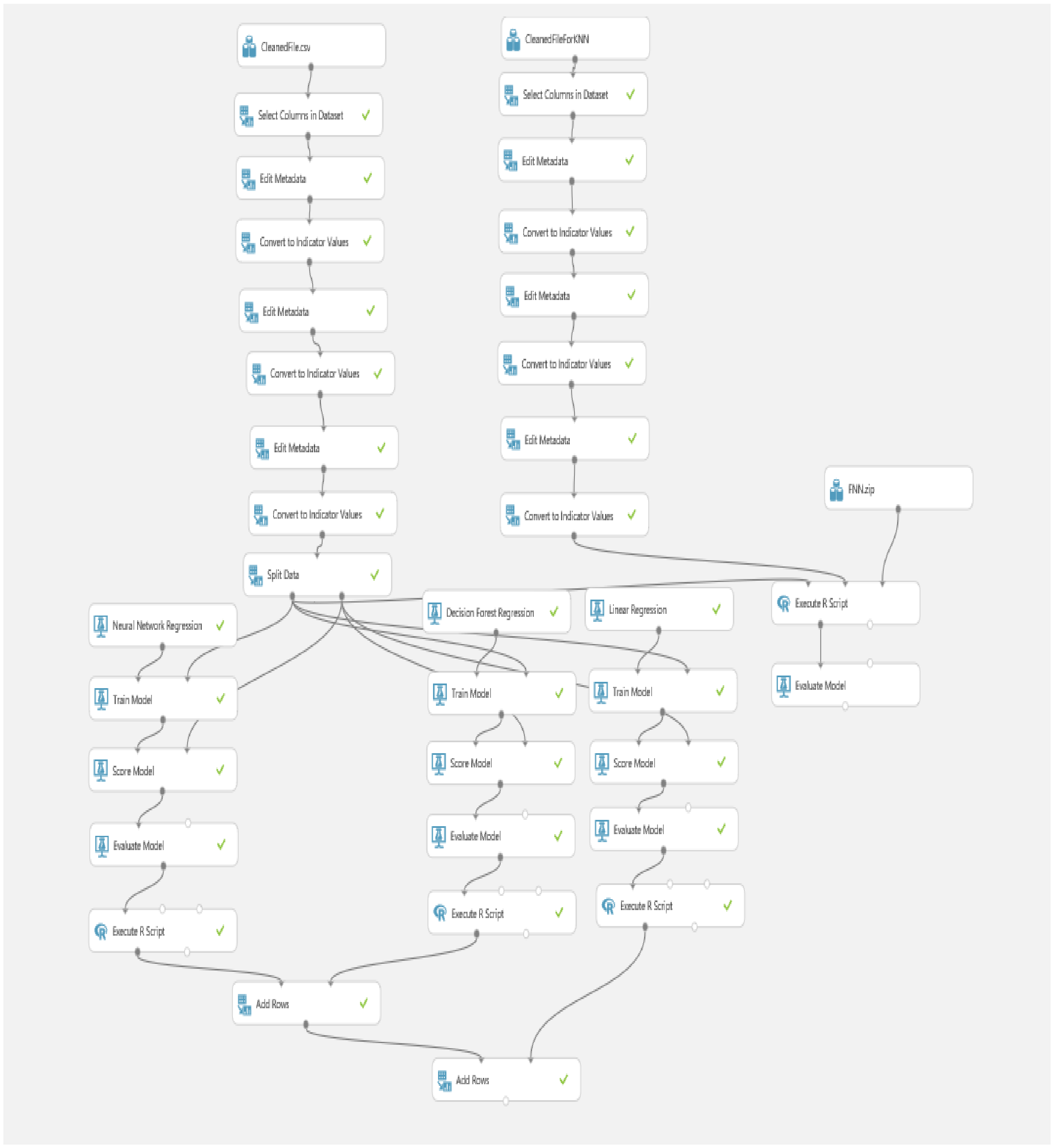
Neural networks are widely known for use in deep learning and modeling complex problems such as image recognition, they are easily adapted to regression problems. Any class of statistical models can be termed a neural network if they use adaptive weights and can approximate non-linear functions of their inputs. Thus neural network regression is suited to problems where a more traditional regression model cannot fit a solution.

RANDOM FOREST REGRESSION

Decision trees are non-parametric models that perform a sequence of simple tests for each instance, traversing a binary tree data structure until a leaf node (decision) is reached.

KNN REGRESSION

There is no in-built component for KNN regression provided my Azure ML studio so we have created a function in Jupyter notebook to check the accuracy of KNN regression model



Assignment2-Prediction Algorithms › Add Rows › Results dataset

rows 3

columns 6

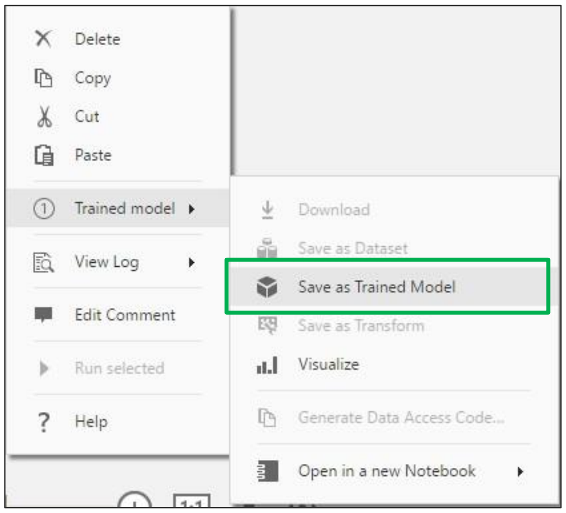
view as

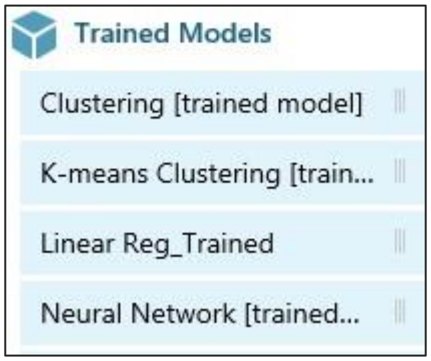
Algorithm	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
Neural Network	0.145084	0.298205	0.040049	0.004257	0.995743
Decision Forest	1.292151	2.045128	0.356686	0.200241	0.799759
Linear Regression	2.229361	2.850165	0.615394	0.388913	0.611087

We choose Neural Network for Predicting Interest rate as its Root Mean Squared Error was less.

Saving TRAINED MODEL

After we run all the models and is satisfied with the Scored Model output, we proceed to save the model by right clicking it.




Trained Models
Clustering [trained model]
K-means Clustering [train...
Linear Reg_Trained
Neural Network [trained...

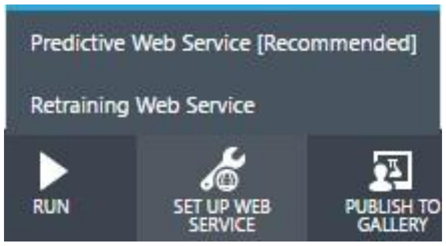
SETUP WEB SERVICE: REGRESSION

CONVERT THE TRAINING EXPERIMENT TO A PREDICTIVE EXPERIMENT

Converting to a predictive experiment involves three steps:

1. Save the model we've trained and then replace our training modules
2. Trim the experiment to remove modules that were only needed for training
3. Define where the Web service will accept input and where it generates the output

All three steps can be accomplished by clicking "Set Up Web service" at the bottom of the experiment.

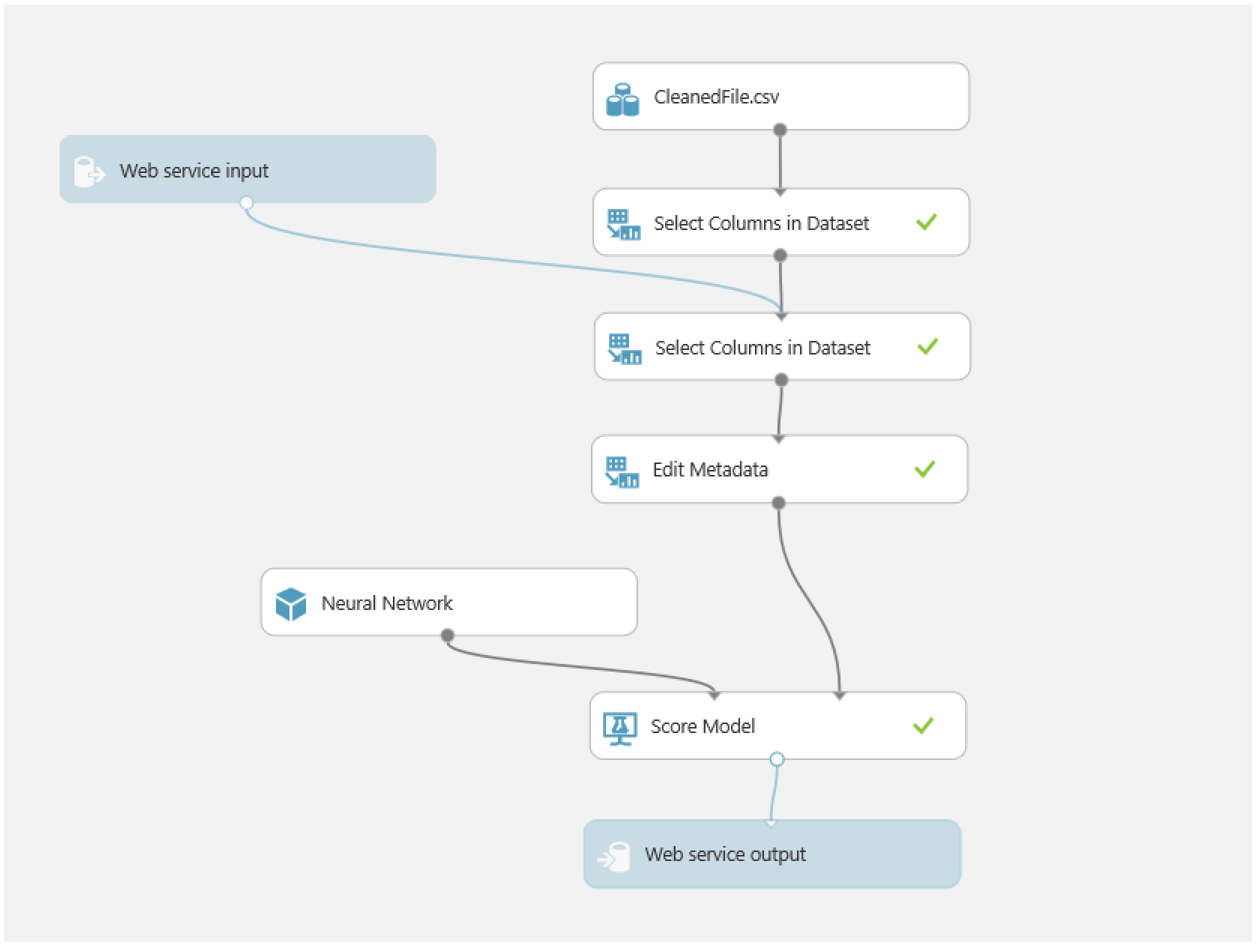


When we click Set Up Web service, several things happen:

The trained model is saved as a single Trained Model module into the module palette to the left of the experiment canvas (we can find it under Trained Models). Modules that were used for training are removed.

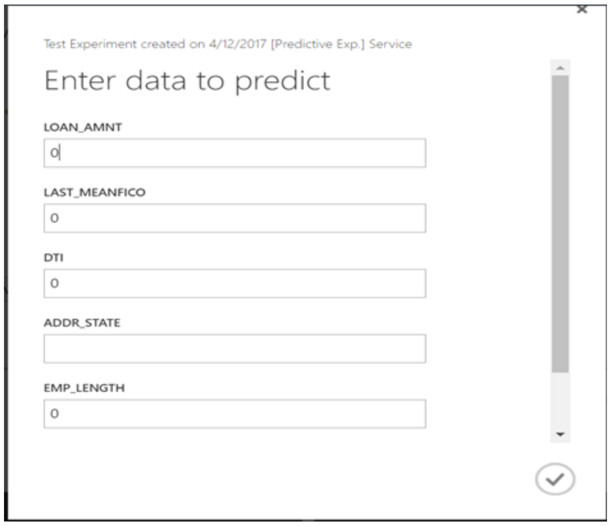
DEPLOY THE WEB SERVICE

We can deploy the experiment as either a classic Web service or a new Web service based on Azure Resource Manager.



To deploy a classic Web service derived from our experiment, click **Deploy Web Service** below the canvas and select **Deploy Web Service [Classic]**. Machine Learning Studio deploys the experiment as a Web service and takes you to the dashboard for that Web service.

From here, you can return to the experiment (**View snapshot** or **View latest**) and run a simple test of the Web service (See **Test the Web service** below).



Test Experiment created on 4/12/2017 [Predictive Exp.] Service

Enter data to predict

LOAN_AMNT	0
LAST_MEANFICO	0
DTI	0
ADDR_STATE	
EMP_LENGTH	0

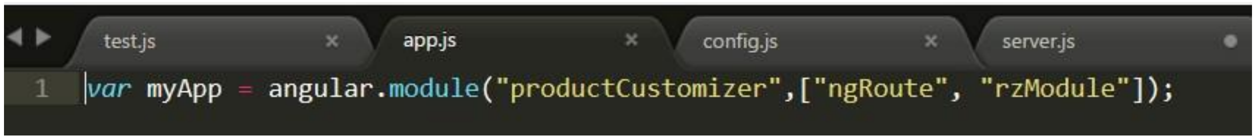
Th results will be a JSON format output.

WEB IMPLEMENTATION

We will be using Angular.js as our frontend framework to call the Rest API for each part. Angular.js has following main component's.

APP.JS

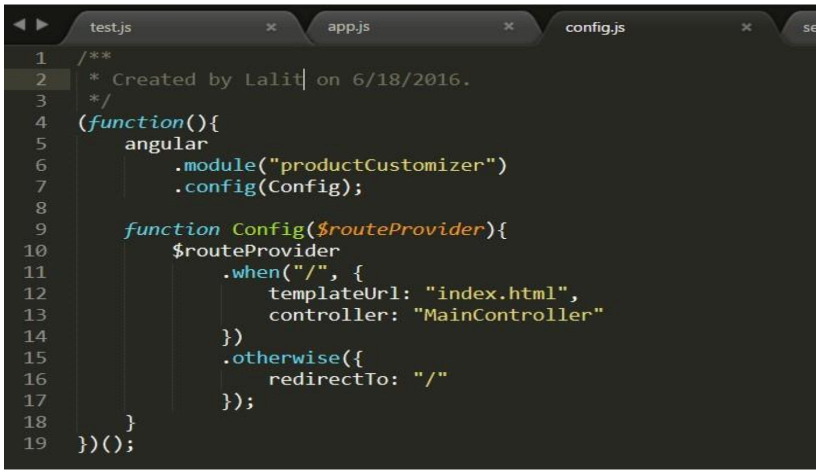
We will define out Application name as shown below



```
var myApp = angular.module("productCustomizer",["ngRoute", "rzModule"]);
```

CONFIG.JS

A Configuration file will need to be made for the routing of the web page. We will define our main page as index.html and a controller which is required by Angular as Main Controller.

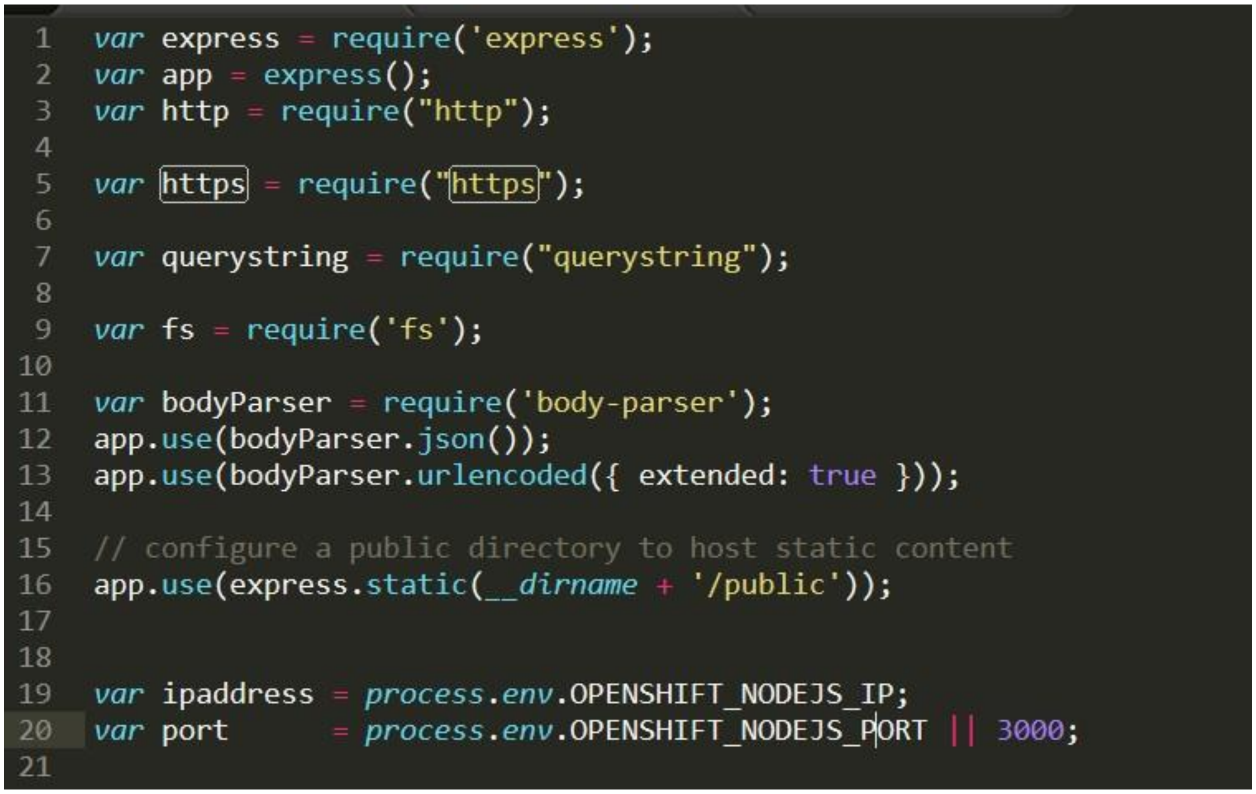


```
/**
 * Created by Lalit on 6/18/2016.
 */
(function(){
    angular
        .module("productCustomizer")
        .config(Config);

    function Config($routeProvider){
        $routeProvider
            .when("/", {
                templateUrl: "index.html",
                controller: "MainController"
            })
            .otherwise({
                redirectTo: "/"
            });
    }
})();
```

SERVER.JS

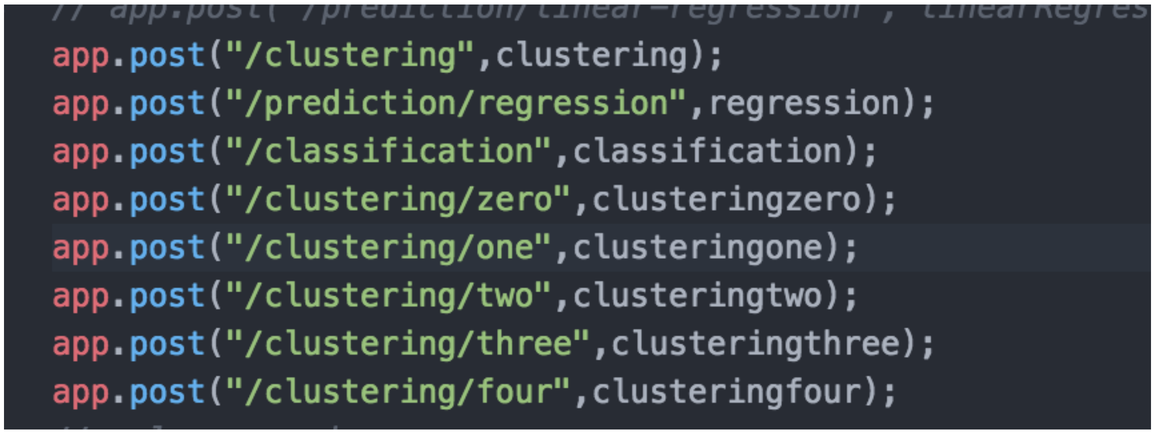
This will be default location for the server control. Application port's, redirection and function's to be called for REST api will be handled here

First, we will load all the required libraries for the HTTP functionality as shown below



```
var express = require('express');
var app = express();
var http = require("http");

var https = require("https");

var querystring = require("querystring");

var fs = require('fs');

var bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// configure a public directory to host static content
app.use(express.static(__dirname + '/public'));


var ipaddress = process.env.OPENSHIFT_NODEJS_IP;
var port      = process.env.OPENSHIFT_NODEJS_PORT || 3000;

```

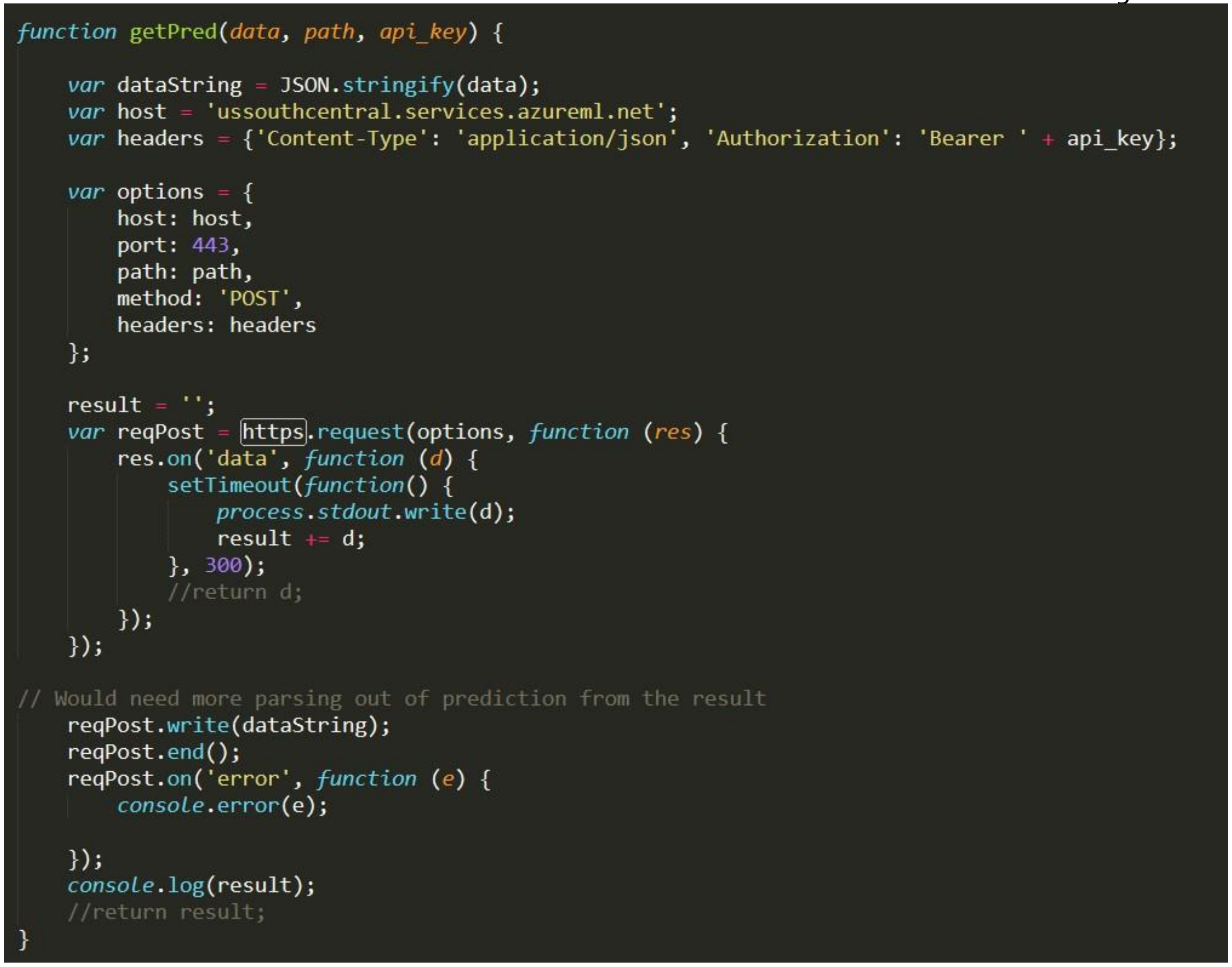
Then, we will define our application's posting url. This is essentially the routing post, where based on the type of prediction we will create different functions



```
// app.post("/prediction/linear-regression", linearRegres
app.post("/clustering",clustering);
app.post("/prediction/regression",regression);
app.post("/classification",classification);
app.post("/clustering/zero",clusteringzero);
app.post("/clustering/one",clusteringone);
app.post("/clustering/two",clusteringtwo);
app.post("/clustering/three",clusteringthree);
app.post("/clustering/four",clusteringfour);
```

So we will use three different post for three different type of predictions that we will perform

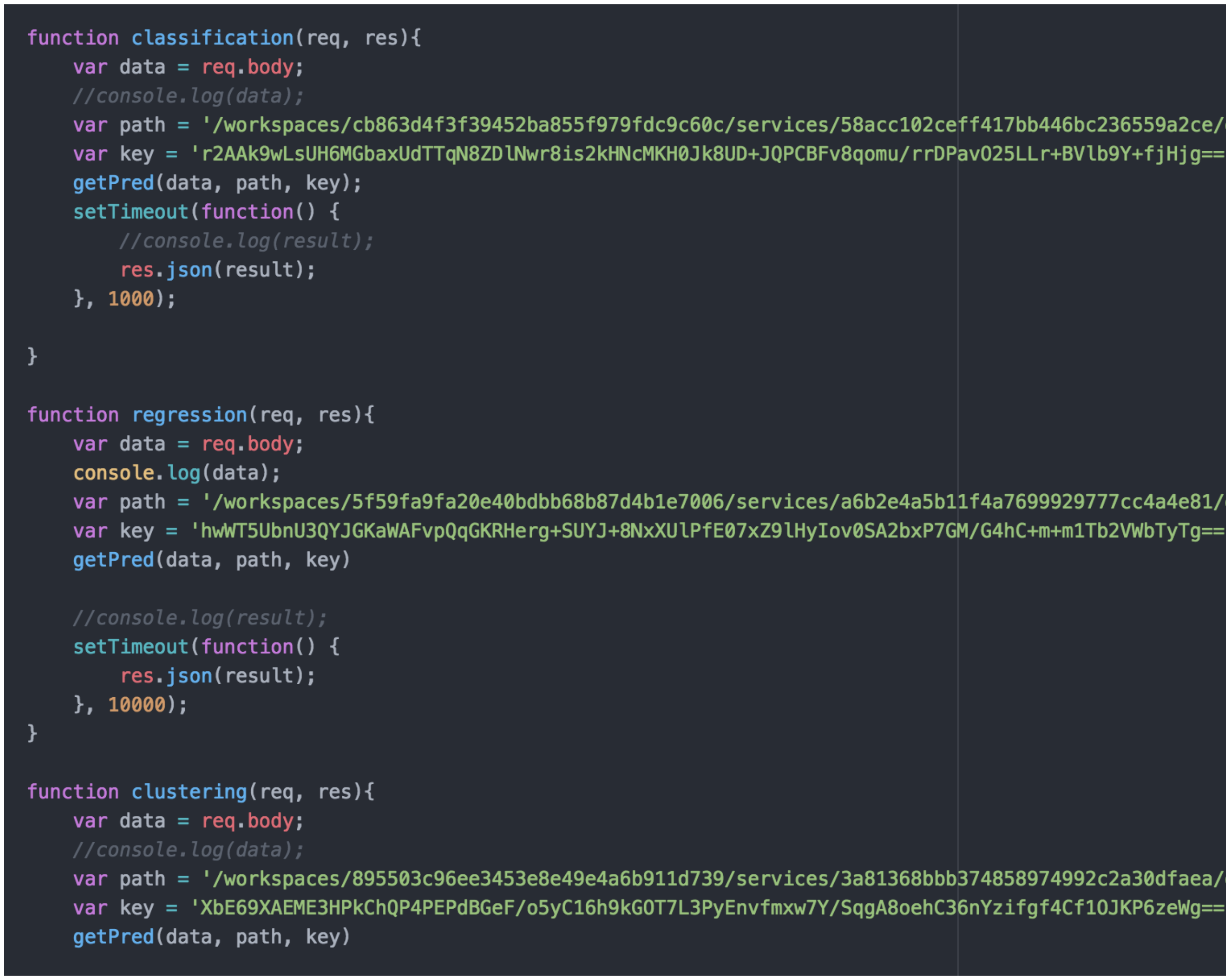
Now we can define our getPred function which we will use to call our api call and get the result back. This function will take data of the user input, api key and url for the call to work.



```
function getPred(data, path, api_key) {

    var dataString = JSON.stringify(data);
    var host = 'ussouthcentral.services.azureml.net';
    var headers = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + api_key};

    var options = {
        host: host,
        port: 443,
        path: path,
        method: 'POST',
        headers: headers
    };

    result = '';
    var reqPost = https.request(options, function (res) {
        res.on('data', function (d) {
            setTimeout(function() {
                process.stdout.write(d);
                result += d;
            }, 300);
            //return d;
        });
    });

// Would need more parsing out of prediction from the result
    reqPost.write(dataString);
    reqPost.end();
    reqPost.on('error', function (e) {
        console.error(e);

    });
    console.log(result);
    //return result;
}
```

Output of the function will be stored in a result variable and can be consumed later.

Now since we have three different calls to make, we will define 3 different functions which will basically provide respective data, api and url to the getPred function defined above as shown below



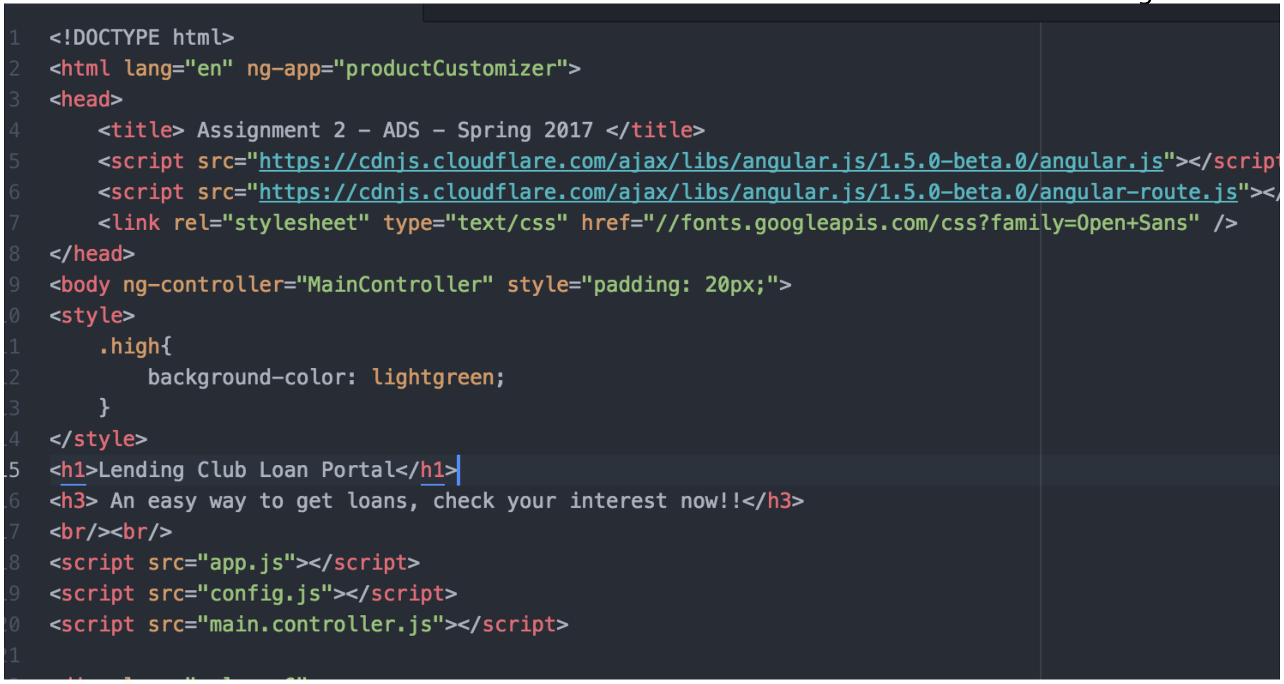
```
function classification(req, res){
    var data = req.body;
    //console.log(data);
    var path = '/workspaces/cb863d4f3f39452ba855f979fdc9c60c/services/58acc102ceff417bb446bc236559a2ce/
    var key = 'r2AAk9wLsUH6MGbaxUdTTqN8ZDlNwr8is2kHNcMKH0Jk8UD+JQPCBFv8qomu/rrDPav025LLr+BVlb9Y+fjHjg==
    getPred(data, path, key);
    setTimeout(function() {
        //console.log(result);
        res.json(result);
    }, 1000);

}

function regression(req, res){
    var data = req.body;
    console.log(data);
    var path = '/workspaces/5f59fa9fa20e40bdbb68b87d4b1e7006/services/a6b2e4a5b11f4a7699929777cc4a4e81/
    var key = 'hwWT5UbnU3QYJGKaWAFvpQqGKRHerg+SUYJ+8NxXUlPfE07xZ9lHyIov0SA2bxP7GM/G4hC+m+m1Tb2VWbTyTg==
    getPred(data, path, key)

    //console.log(result);
    setTimeout(function() {
        res.json(result);
    }, 10000);
}

function clustering(req, res){
    var data = req.body;
    //console.log(data);
    var path = '/workspaces/895503c96ee3453e8e49e4a6b911d739/services/3a81368bbb374858974992c2a30dfaea/
    var key = 'XbE69XAEME3HPkChQP4PEPdBGeF/o5yC16h9kGOT7L3PyEnvfmxw7Y/SqgA8oehC36nYzifgf4Cf10JKP6zeWg==
    getPred(data, path, key)
```

Now before we jump to MainContoller, we will design our web page. Since, MainController require elements fetch from the front end, we will define the home page and use those element's in the MainController later

INDEX.HTML

We will say the home page to use our app defined by ng-app. in the main html tag. Similarly, we will add scripting components to use required angular.js libraries.

We will provide ng-controller "Main Controller" to the main body section, which we will define later. We will also add custom script's we defined earlier (app.js,config.js and main.controller.js)



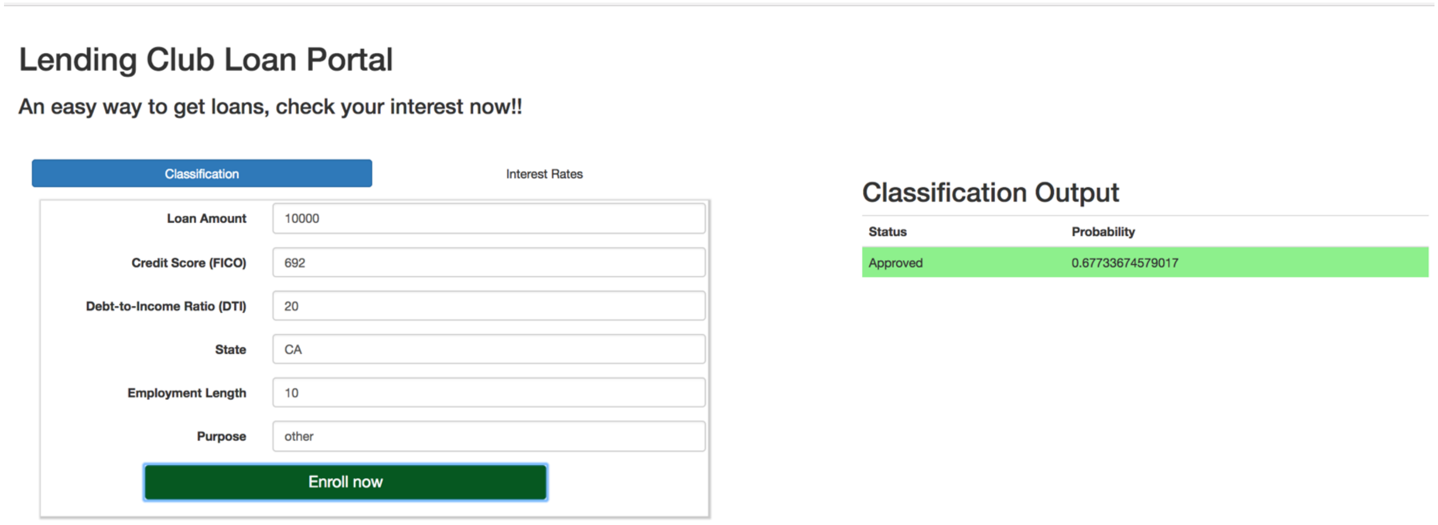
```
<!DOCTYPE html>
<html lang="en" ng-app="productCustomizer">
<head>
    <title> Assignment 2 - ADS - Spring 2017 </title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.5.0-beta.0/angular.js"></scrip
    <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.5.0-beta.0/angular-route.js"></
    <link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Open+Sans" />
</head>
<body ng-controller="MainController" style="padding: 20px;">
<style>
    .high{
        background-color: lightgreen;
    }
</style>
<h1>Lending Club Loan Portal</h1>
<h3> An easy way to get loans, check your interest now!!</h3>
<br/><br/>
<script src="app.js"></script>
<script src="config.js"></script>
<script src="main.controller.js"></script>
```

For the user control on the type of Prediction to be used, will provide 3 button's and link those button's to ng-method's to distinguish and control the flow.

For the type of features we decided for each type of Prediction, we will use div components and tag them with related ng-method. For example, if we want to use Temperature as a component for Regression, we will mark the div component as ng-method == "Prediction" which we defined earlier.

So on and so forth respectively for each type of Prediction method. We can see the result of the HTML page below

Output of the HTML page can be shown as below:



Lending Club Loan Portal

An easy way to get loans, check your interest now!!

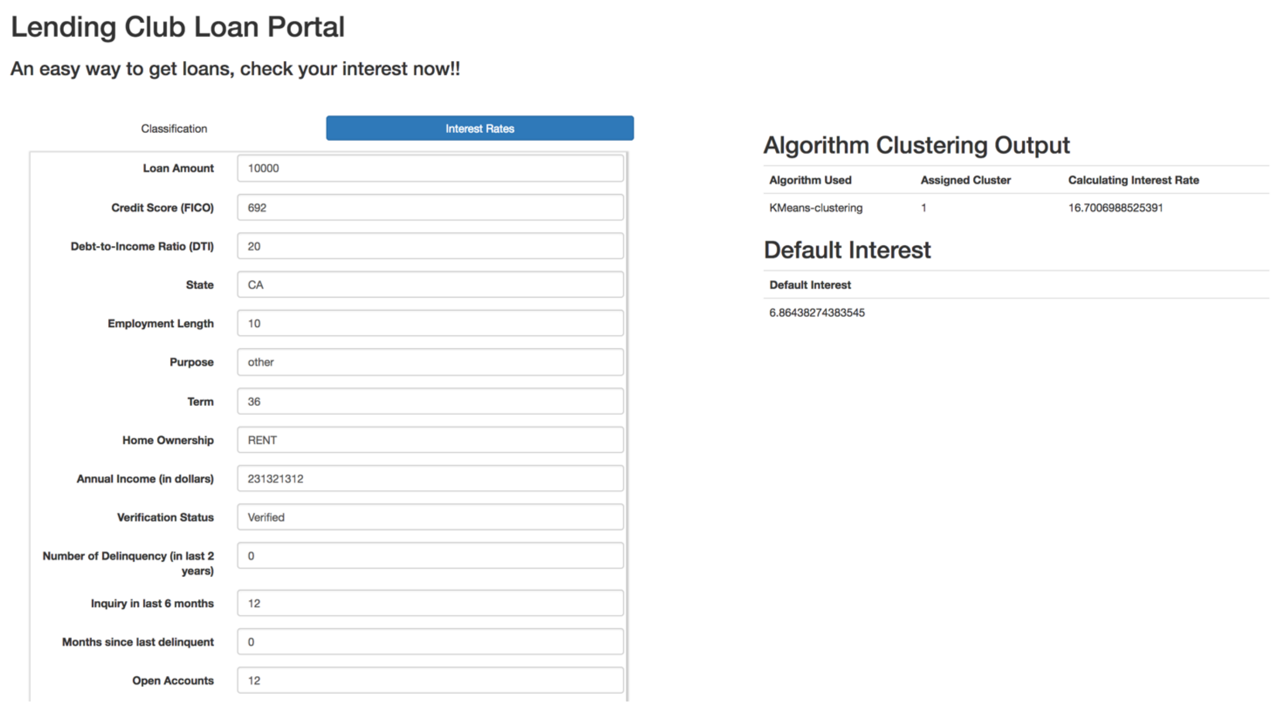
Loan Amount	10000
Credit Score (FICO)	692
Debt-to-Income Ratio (DTI)	20
State	CA
Employment Length	10
Purpose	other

Classification Output

Status	Probability
Approved	0.67733674579017

As we can see that, defending on the type of method selected and our features required for those prediction method's we will show the user those input's only.

Similarly, on the right hand side, will display the result. Again depending on the type of method selected, we will change the output table



Lending Club Loan Portal

An easy way to get loans, check your interest now!!

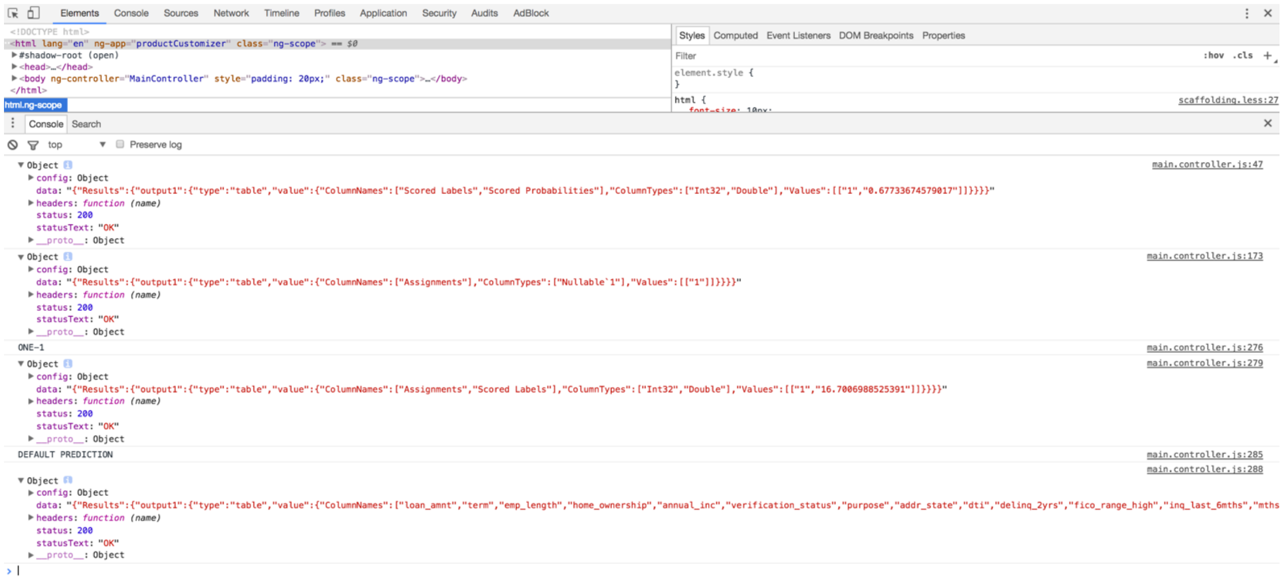
Loan Amount	10000
Credit Score (FICO)	692
Debt-to-Income Ratio (DTI)	20
State	CA
Employment Length	10
Purpose	other
Term	36
Home Ownership	RENT
Annual Income (in dollars)	231321312
Verification Status	Verified
Number of Delinquency (in last 2 years)	0
Inquiry in last 6 months	12
Months since last delinquent	0
Open Accounts	12

Algorithm Clustering Output

Algorithm Used	Assigned Cluster	Calculating Interest Rate
KMeans-clustering	1	16.7006988525391

Default Interest

Default Interest
6.86438274383545



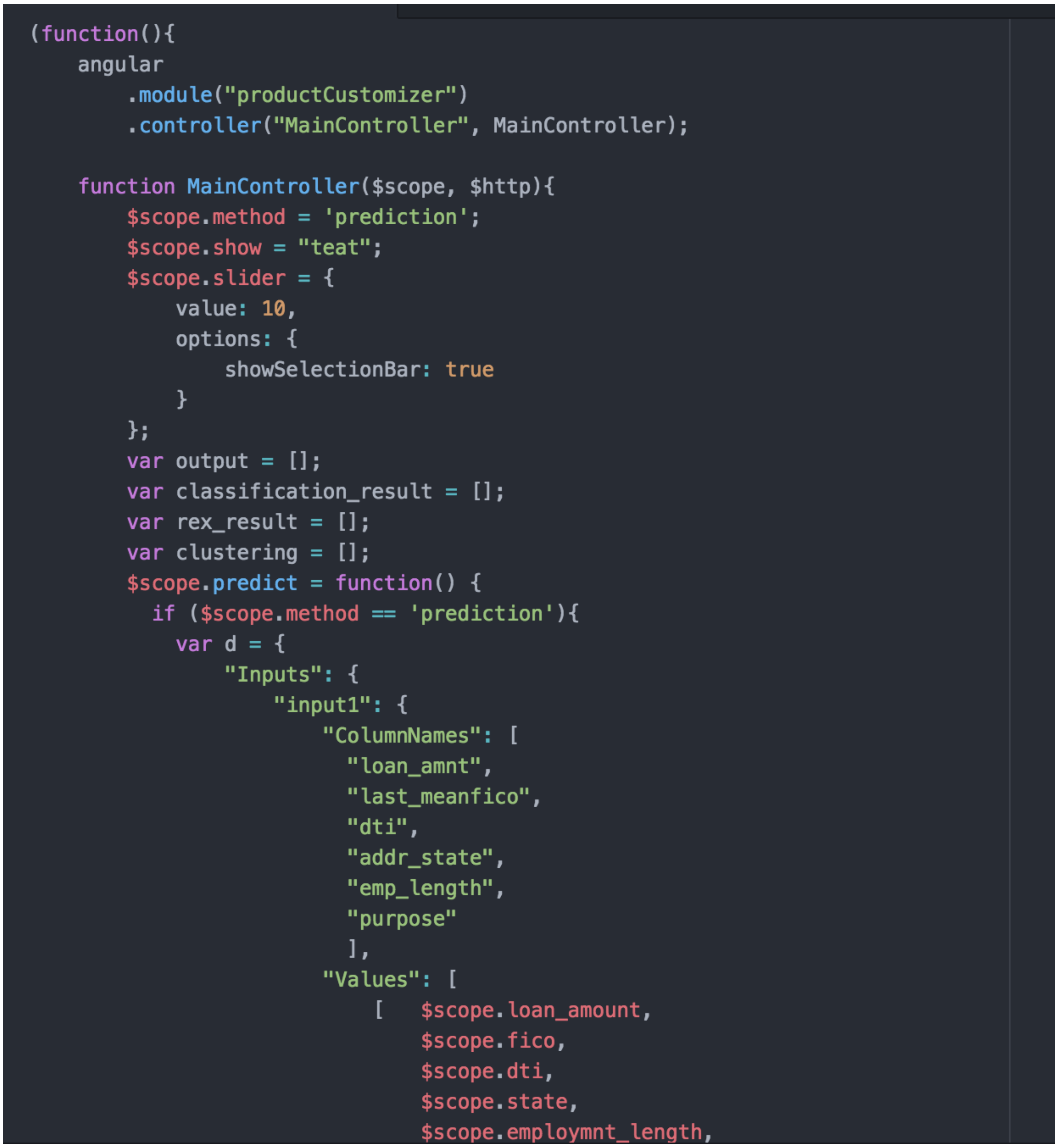
```
<!DOCTYPE html>
<html lang="en" ng-app="productCustomizer" class="ng-scope"> == $0
  ▶#shadow-root (open)
  ▶<head>…</head>
  ▶<body ng-controller="MainController" style="padding: 20px;" class="ng-scope">…</body>
</html>
html.ng-scope
```

```
element.style {
}
html {                                                    scaffolding.less:27
  font-size: 10px;
```

```
▼Object                                                                                    main.controller.js:47
  ▶config: Object
   data: "{"Results":{"output1":{"type":"table","value":{"ColumnNames":["Scored Labels","Scored Probabilities"],"ColumnTypes":["Int32","Double"],"Values":[["1","0.67733674579017"]]}}}}"
  ▶headers: function (name)
   status: 200
   statusText: "OK"
  ▶__proto__: Object
▼Object                                                                                    main.controller.js:173
  ▶config: Object
   data: "{"Results":{"output1":{"type":"table","value":{"ColumnNames":["Assignments"],"ColumnTypes":["Nullable`1"],"Values":[["1"]]}}}}"
  ▶headers: function (name)
   status: 200
   statusText: "OK"
  ▶__proto__: Object
ONE-1                                                                                      main.controller.js:276
▼Object                                                                                    main.controller.js:279
  ▶config: Object
   data: "{"Results":{"output1":{"type":"table","value":{"ColumnNames":["Assignments","Scored Labels"],"ColumnTypes":["Int32","Double"],"Values":[["1","16.7006988525391"]]}}}}"
  ▶headers: function (name)
   status: 200
   statusText: "OK"
  ▶__proto__: Object
DEFAULT PREDICTION                                                                         main.controller.js:285
                                                                                           main.controller.js:288
▼Object
  ▶config: Object
   data: "{"Results":{"output1":{"type":"table","value":{"ColumnNames":["loan_amnt","term","emp_length","home_ownership","annual_inc","verification_status","purpose","addr_state","dti","delinq_2yrs","fico_range_high","inq_last_6mths","mths
  ▶headers: function (name)
   status: 200
   statusText: "OK"
  ▶__proto__: Object
> |
```

Output table can be seen as shown

MAIN CONTROLLER



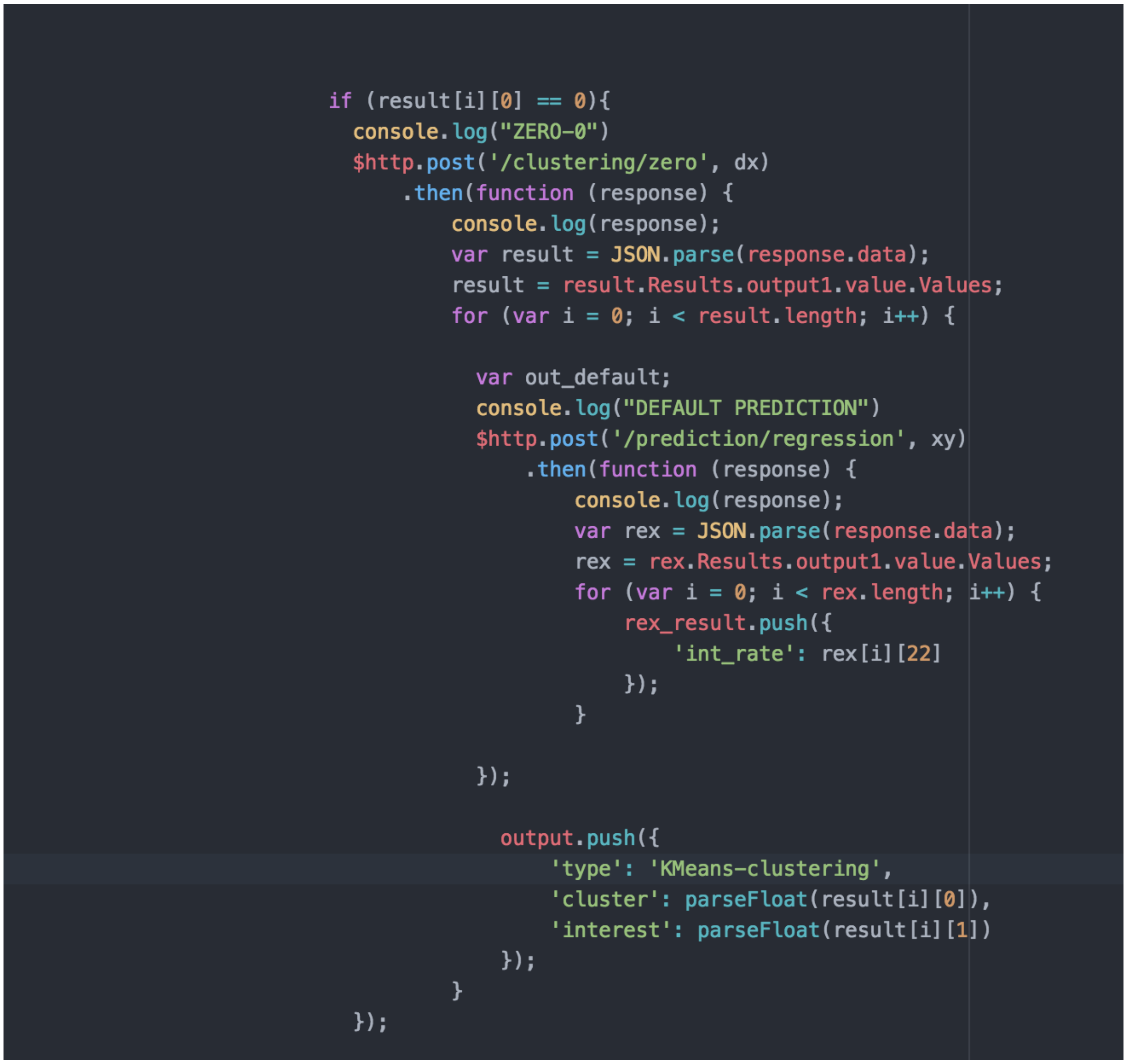
```
(function(){
    angular
        .module("productCustomizer")
        .controller("MainController", MainController);

    function MainController($scope, $http){
        $scope.method = 'prediction';
        $scope.show = "teat";
        $scope.slider = {
            value: 10,
            options: {
                showSelectionBar: true
            }
        };
        var output = [];
        var classification_result = [];
        var rex_result = [];
        var clustering = [];
        $scope.predict = function() {
          if ($scope.method == 'prediction'){
            var d = {
                "Inputs": {
                    "input1": {
                        "ColumnNames": [
                          "loan_amnt",
                          "last_meanfico",
                          "dti",
                          "addr_state",
                          "emp_length",
                          "purpose"
                          ],
                        "Values": [
                            [   $scope.loan_amount,
                                $scope.fico,
                                $scope.dti,
                                $scope.state,
                                $scope.employmnt_length,
```

Now since we have defined the ng-tag of the input's in the HTML page, we can use those tag's in the Main controller to fetch the data and pass to the server.js to get the output.

Here we are creating different variables like Output for Regression Output, Classification_result for classification results and so on.

Now depending on the method used by the user we will use different functions, format for the data input style can be seen from the Request-Response API documentation in the Azure studio. Scope will be used to fetch the elements form the front end.

Now since we have everything we need, we can call our server.js with all the data that we have and using the appropriate app post. Output of the result will be saved and will be displayed in the HTML page. Similarly, we can add additional functions for Classification and clustering model's based on the user selection and using its respective data format.



```
if (result[i][0] == 0){
  console.log("ZERO-0")
  $http.post('/clustering/zero', dx)
      .then(function (response) {
          console.log(response);
          var result = JSON.parse(response.data);
          result = result.Results.output1.value.Values;
          for (var i = 0; i < result.length; i++) {

            var out_default;
            console.log("DEFAULT PREDICTION")
            $http.post('/prediction/regression', xy)
                .then(function (response) {
                    console.log(response);
                    var rex = JSON.parse(response.data);
                    rex = rex.Results.output1.value.Values;
                    for (var i = 0; i < rex.length; i++) {
                        rex_result.push({
                            'int_rate': rex[i][22]
                        });
                    }

            });

              output.push({
                  'type': 'KMeans-clustering',
                  'cluster': parseFloat(result[i][0]),
                  'interest': parseFloat(result[i][1])
              });
          }
  });
```

FINAL CODE

It can be found on github's link submitted in the email.