House Sale Price Prediction

Alyssa Peterson

Sriram Ramadoss Venkata

Heather Simmons

Jessica Urban

Michael Xiong

Agenda

- Introduction
- About Dataset
- Linear Regression
- Neural Networks
- Random Forest
- Support Vector Machine
- Gaussian Mixture Model
- Algorithm Comparisons
- Q & A

Introduction

- Our goal for this project was to use regression and classification techniques in order to estimate the sale price of a house in King County, Washington given the feature and pricing data for around 21,000 houses sold within one year.

About Dataset

- Our dataset comes from a Kaggle competition.
- Our dataset contains house sale prices and its features for homes sold in King County, Washington between May 2014 and May 2015.
- King County is the most populous county in Washington and is included in the Seattle-Tacoma-Bellevue metropolitan statistical area. The county is considered the 13^{th} most populous county in the United States.
- There are 21,613 observations in the dataset.
- There are 25 total attributes in the dataset, four of which we derived from current columns. We are planning to use 22 attributes in our models: all attributes except for date, latitude and longitude.

Dataset-Preprocessing

- During cleaning and preprocessing, we created four attributes derived from other attributes: Age, Age_revovated, Sqrt_living15_diff, and Sqft_lot15_diff.
- We chose not to use the variable data, because it only shows us when the data was entered into the database.
- We chose not to use latitude and longitude because the attribute, Zipcode, contains the same information and was easier to work with in our models.
- We checked for missing variables and the dataset didn't contain any.
- We performed feature selection by looking at the correlation percentage of each attribute with price.

Attribute	Percentage
Bedrooms	0.308349598
Bathroom	0.525137505
Sqft_living	0.702035055
Sqft_lot	0.089660861
Floors	0.256793888
Waterfront	0.266369434
View	0.397293488
Condition	0.036361789
Grade	0.667434256
Sqft_above	0.605567298
Sqft_basement	0.323816021
Yr_built	0.054011531
Age	-0.054011531
Yr_renovated	0.126433793
Age_renovation	-0.105754631
Sqft_living15	0.585378904
Sqft_living15_diff	0.405391664
Sqft_lot15	0.082447153
Sqft_lot15_diff	0.050590661

Outlier Detection using Cook’s Distance

- We used Cook's distance for the original dataset and found one outlier. It was removed to create dataset 2.
- After applying Cook's distance to dataset 2, we found 608 outliers. They were removed to create dataset 3.
- The team used dataset 2 and dataset 3 for the project.

Linear Regression

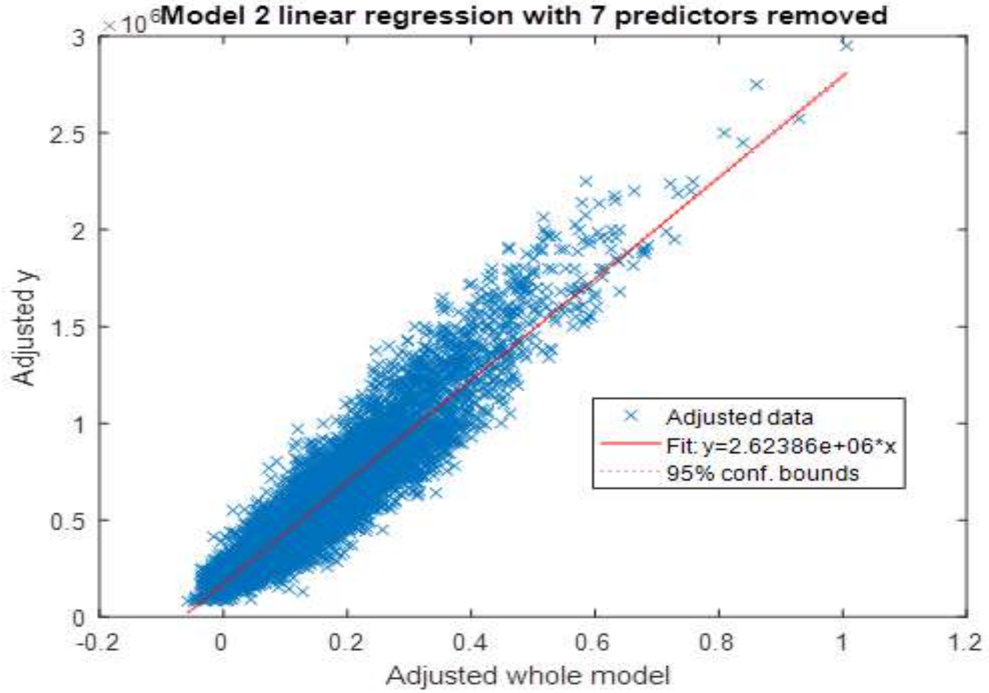
- Used MATLAB to create 3 models
- Datasets : divided into 80% training, 20% test
  - Dataset 2 - one outlier removed
  - Dataset 3 – 608 outliers removed
- Algorithm
  - Model 1 used dataset 2
  - Model 2 used dataset 3 with 7 predictors removed
  - Models 3 used dataset 3 with 15 predictors removed
  - (yhat= predict(mdl,Xnew))

Linear Regression

Analytics Results:

Model 2 Training Metrics:
Root Mean Squared Error: $96,800
R-squared: 0.879
Adjusted R-Squared 0.878

Model 2 Test Metrics:
Root Mean Squared Error: $94,927.38
R-squared: 0.913223143



Neural Networks

- Used Matlab's built in functions: fitnet and feedforwardnet
- Tried two different methods: Levenberg-Marquardt and Bayesian
- Normalized with mapminmax to scale the targets, where the output of the network will be trained to produce outputs in the range [–1,+1]
- Started with 1 hidden layer and 10 neurons.
- Ran each combination again with 100 hidden neurons to determine which one would perform better.

Neural Networks

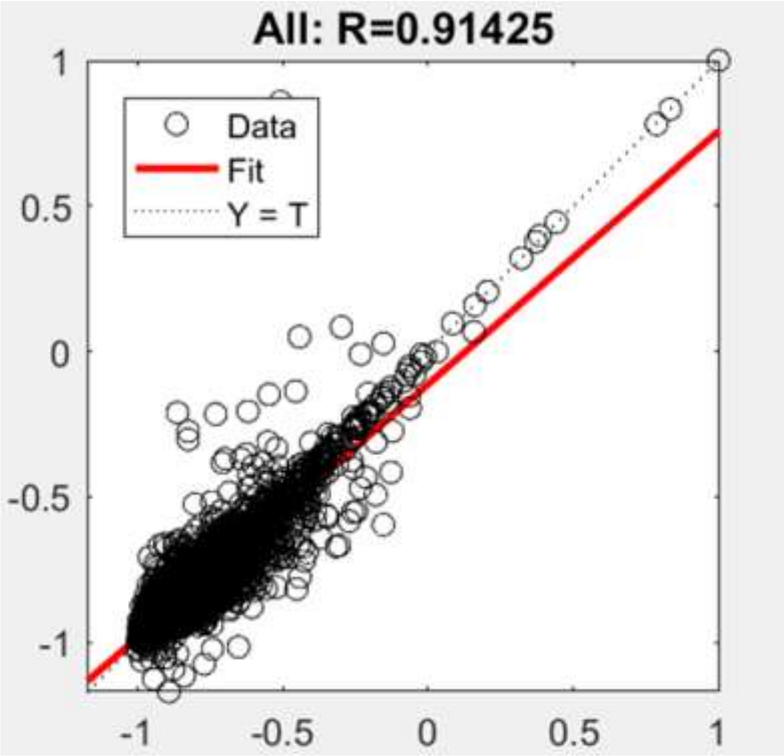
Analytics Results:

10 neurons	MSE	R-value	slope	y-intercept	MAE
FeedForward Bayesian	0.0019	0.8941	0.8185	-0.1595	0.0289
FitNet Bayesian	0.0018	0.8951	0.8064	-0.1701	0.0295
FitNet LevenbergMarquardt	0.0019	0.8931	0.797	-0.1785	0.0294
FeedForward LevenbergMarquardt	0.0021	0.8826	0.7863	-0.1865	0.0304
100 neurons	**MSE**	**R-value**	**slope**	**y-intercept**	**MAE**
FeedForward Bayesian	0.0019	0.8983	0.8922	-0.0944	0.025
FitNet Bayesian	0.0015	0.9142	0.8715	-0.1128	0.0258
FitNet LevenbergMarquardt	0.0017	0.9009	0.8209	-0.1574	0.0283
FeedForward LevenbergMarquardt	0.002	0.8878	0.8053	-0.1714	0.03

Neural Networks

Analytics Results:

- **R-Squared score, Root Mean Squared Error (RMSE), Mean Absolute Error** of each model were calculated to find the quality and performance of the algorithm.
- **Fit Net Bayesian** with 100 hidden neurons performed the best
- Took nearly 2 hours to run the model
- R-Squared = 0.9142
- RMSE = 0.0015
- Mean Absolute Error = 0.0258

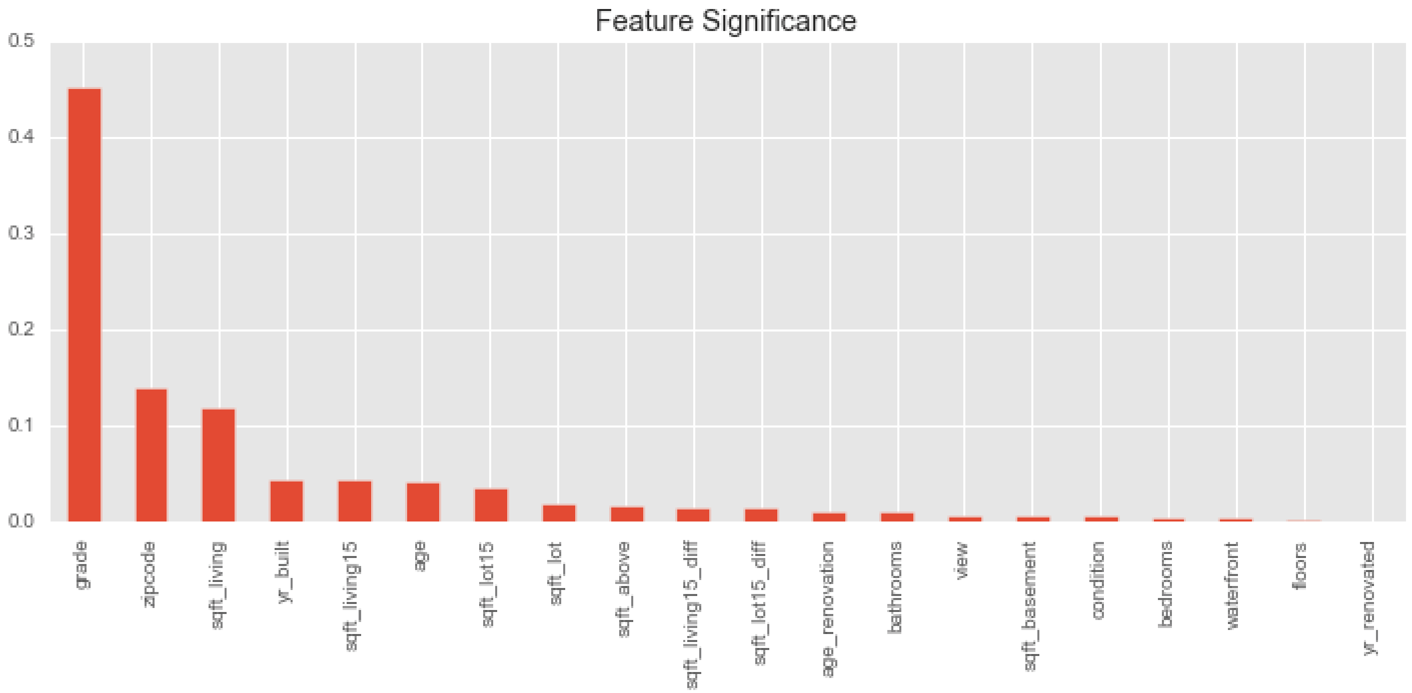


Random Forest

- Written in and executed in Python due to it's extensive sklearn package.
- In this algorithm:
  - Transformed the target variable(price) to the log to correct for skew
  - Removed outliers using a function called median absolute deviation
  - Calculated the skew for each variable
  - Transformed variables if the skew was greater than 0.75
  - Applied a normalizer to each column
  - Ran Random Forest using 10 fold cross validation

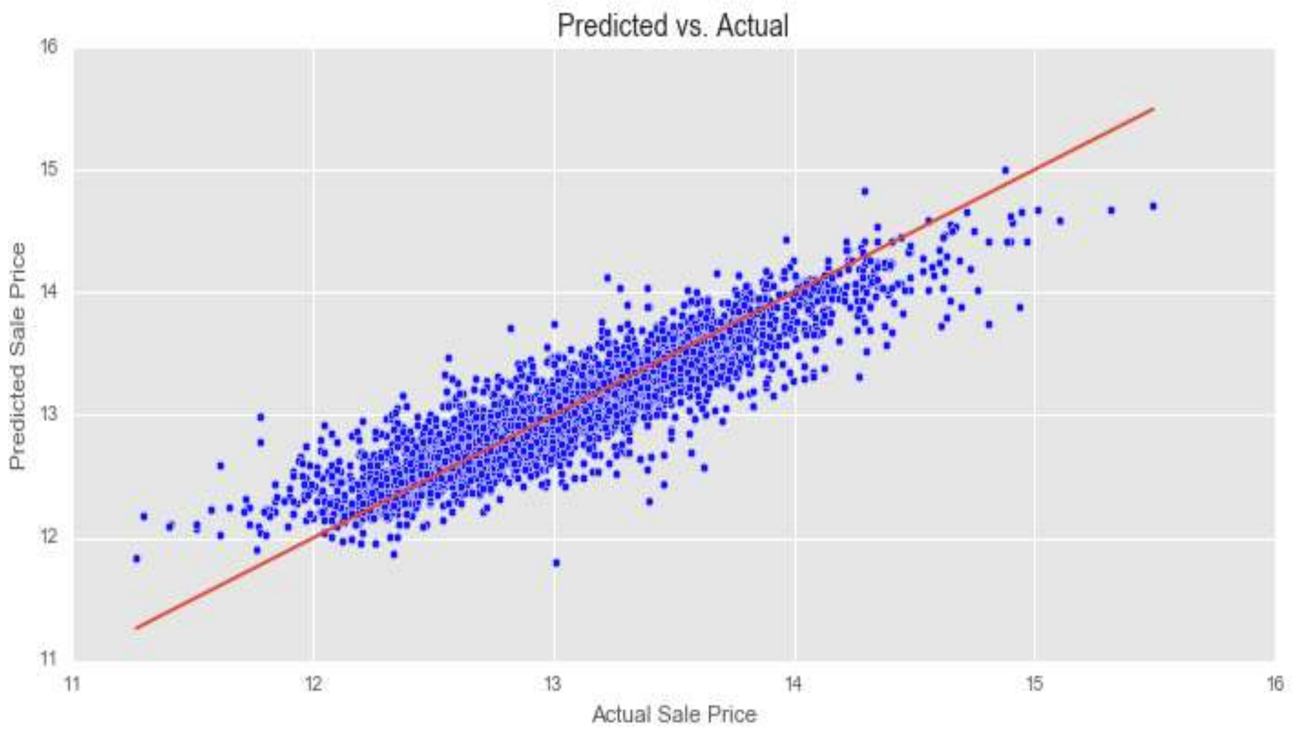
Random Forest

- **Feature Selection:**
- Outputted feature importance coefficients, mapped them to their feature name, and sorted in decreasing order. Given our choice of model and methods for preprocessing data the most significant features are: 1. Grade 2. Zipcode 3. Sqft_living 4. Yr_built



Random Forest

- <u>**Analytics Results:**</u>
- **R-Squared score, Root Mean Squared Error (RMSE), Mean Absolute Error, and Explained Variance** of each model were calculated to find the quality and performance of the algorithm.
- **Model 1 performed the best**
- |R-Squared = 0.825 | Adj. R-Squared = 0.825| RMSE = 0.217 |Mean Absolute Error | = 0.157| Explained Variance Score= 0.825



SVM

- A **Support Vector Machine** (SVM) is a discriminative classifier formally defined by a separating hyperplane. Given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

Advantages:

- SVMs produces large margin separating hyperplane, and efficient in higher dimension
- It maximizes the margin between points closest to the boundary
- SVMs only consider points near the margin (support vectors) – more robust

Disadvantages:

- Due to complexity of the algorithm it requires high amount of memory and takes long time to train the model and predict the test data
- The model is sensitive to optimal choice of kernel and regularization parameters

SVM

Data Preparation

- Used Correlation to figure out which predictor contribute more in prediction of prices
- Normalized all predictor to equal scale.
- Converted the target (Price – numerical data) to categorical values and into three bins.

Bin 1	0-300000
Bin2	300000-700000
Bin 3	700000+

SVM

Building Model

```
[rows useless] = size(X);
G_Mat = zeros(rows, rows);
rec = 0;
for i = 1 : rows
    d1 = X(i, :); % get one data point
    rec = rec + 1; col = 0;
    for j = 1 : rows
        d2 = X(j, :); % get another data point
        col = col + 1;
        G_Mat(rec, col) = (1 + d1 * d2')^2; %
    end
end
```

Creating Gram matrix from training data

```
CVMdl = fitcecoc(G_Mat,Y,'ClassNames',classOrder);
CMdl = CVMdl.Trained{1};
```

Fit the model using gram matrix

```
T_newX = [(1 + New_X * X').^2'] ;
[j, score] = predict(CVMdl, T_newX');
```

Predict the New X

SVM

Linear Support Vector Machine

	Predicted – Class 1	Predicted – Class 2	Predicted – Class 3	Total Actual
Actual – Class 1	0	0	0	0
Actual – Class 2	4427	12582	4205	21214
Actual – Class 3	10	182	207	399
Total Predicted	4437	12764	4412	21613

Accuracy = 0.58

Time taken = 10 mins

Kernel – Gaussian 14d

SVM

	Predicted – Class 1	Predicted – Class 2	Predicted – Class 3	Total Actual
Actual – Class 1	839	2197	330	3366
Actual – Class 2	3510	6705	1880	12095
Actual – Class 3	88	3142	2202	5432
Total Predicted	4437	12044	4412	20893

Accuracy = 0.46

Time taken = 1 hour

Kernel - Gaussian Infinite Dimension

SVM

	Predicted – Class 1	Predicted – Class 2	Predicted – Class 3	Total Actual
Actual – Class 1	1548	2881	8	4437
Actual – Class 2	678	11473	613	12764
Actual – Class 3	5	1572	2835	4412
Total Predicted	2231	15376	3456	21613

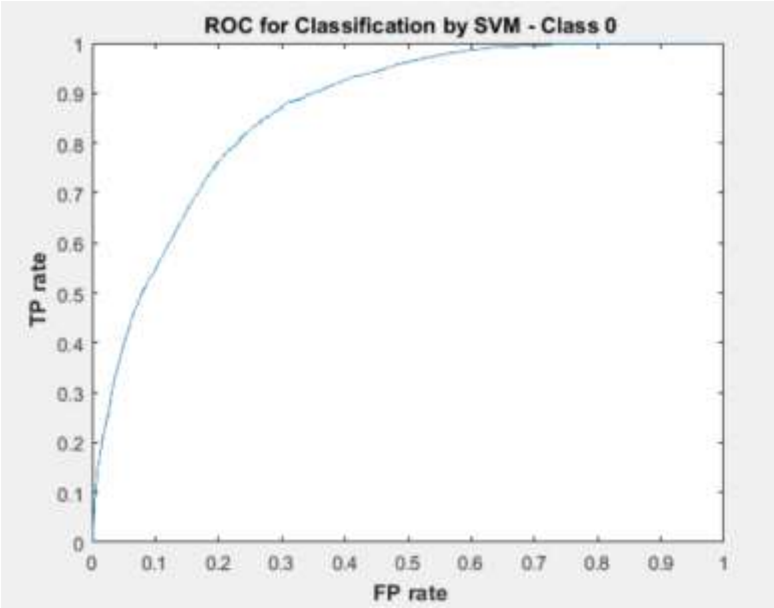
Accuracy = 0.73

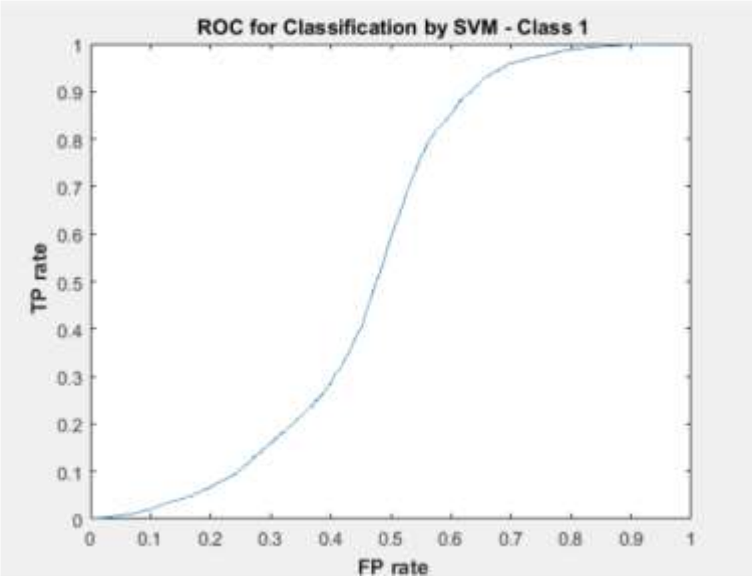
Time taken = 2.5 hours

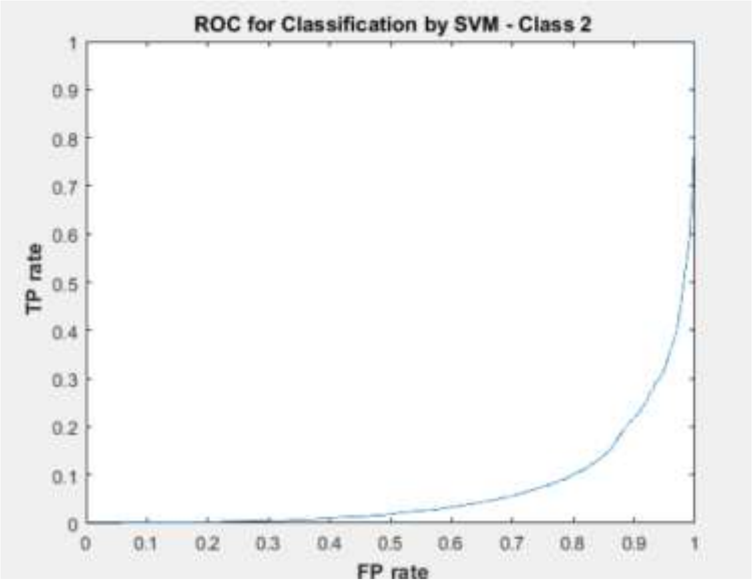
	Class 1	Class 2	Class 3
Precision	0.53	0.69	0.57
Recall	0.69	0.52	0.35
AUC	0.86 58	0.5426	0.20

Kernel - Gaussian Infinite Dimension

ROC







Gaussian Mixture Model

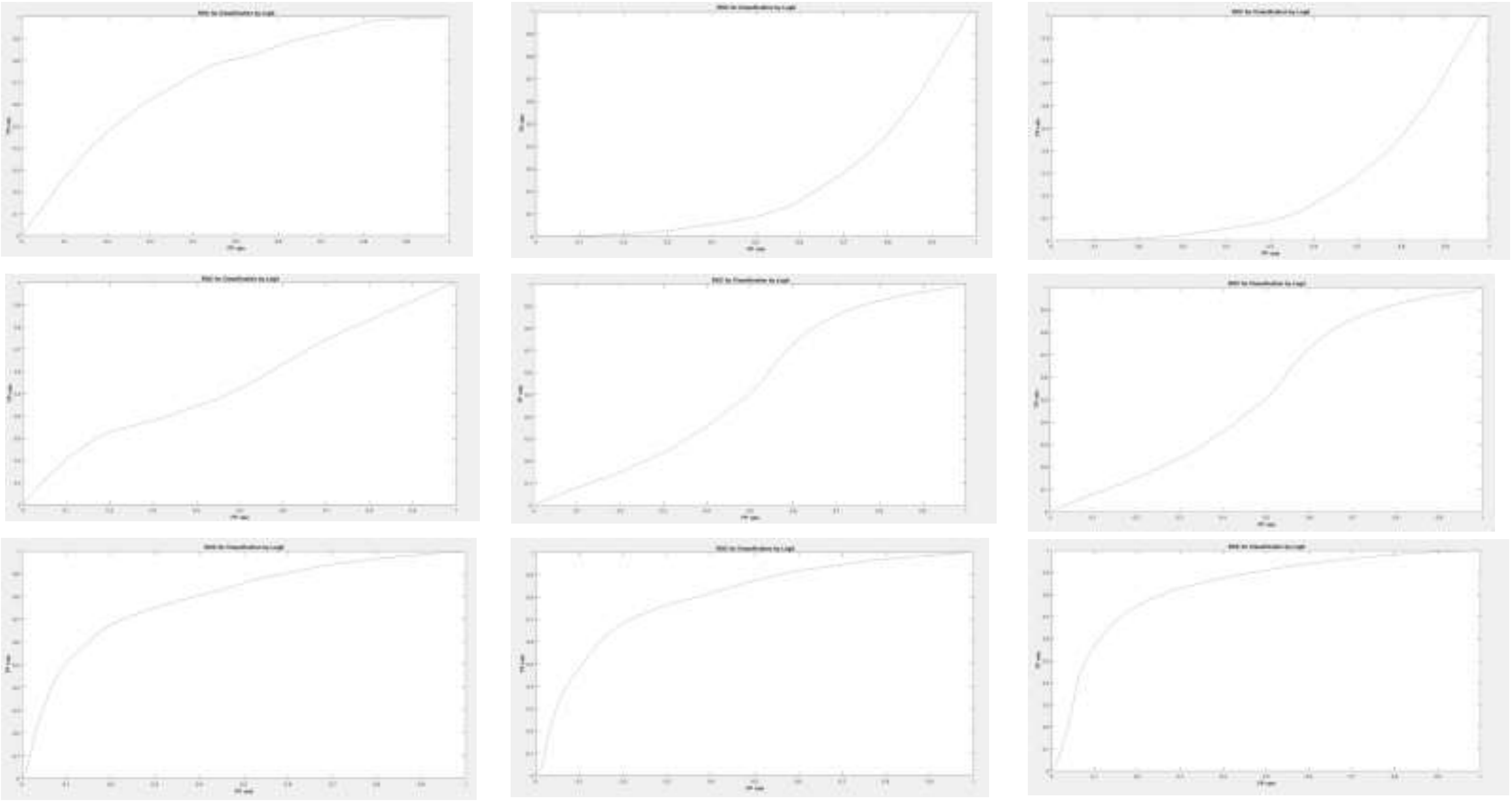
- Bucketed price values into three different groups:
  - Less than or equal to 300K (21.2%);
  - Greater than 300K
  - Less than 700K (58.4%); 700K or greater (20.4%).
- Principle Component Analysis was performed to select the best features.
  - Transformed data into a set of linearly uncorrelated variables.
  - Chose the two components that accounted for the majority of data variance.
- Ran with 1-3 clusters with different levels of regularization.
- Selected clusters with lowest Negative Log-Likelihood.
- Used posterior probability as a predictor to see if this improved the accuracy.
- The best overall model was with 3 clusters and 0 regularization.
- The best model accuracy and lowest negative log-likelihood was with the posterior probability with 0.05 regularization

Gaussian Mixture Model

Class 1:

Class 2:

Class 3:



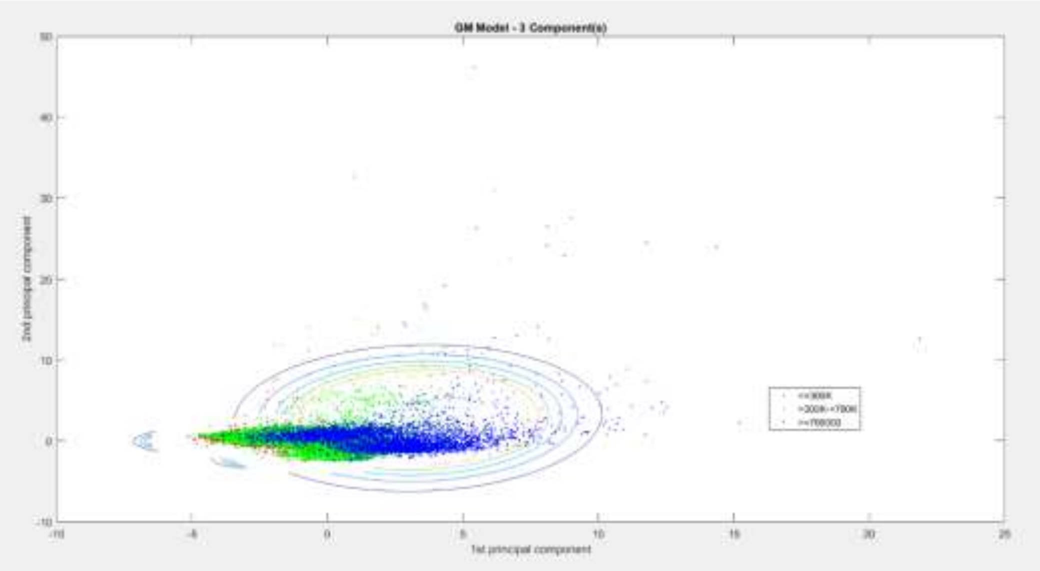
	Model 1			Model 2			Model 3		
NLL	6.85E+04			6.98E+04			-6.25E+03		
Accuracy	0.4928			0.5456			0.5483		
Class	One	Two	Three	One	Two	Three	One	Two	Three
Recall	0.65	0.58	0.08	0.006	0.92	0.04	0.005	0.92	0.039
Precision	0.35	0.59	0.56	0.01	0.62	0.45	0.009	0.62	0.44
AUC	0.7176	0.5517	0.7943	0.2297	0.5316	0.8001	0.2337	0.5318	0.8339

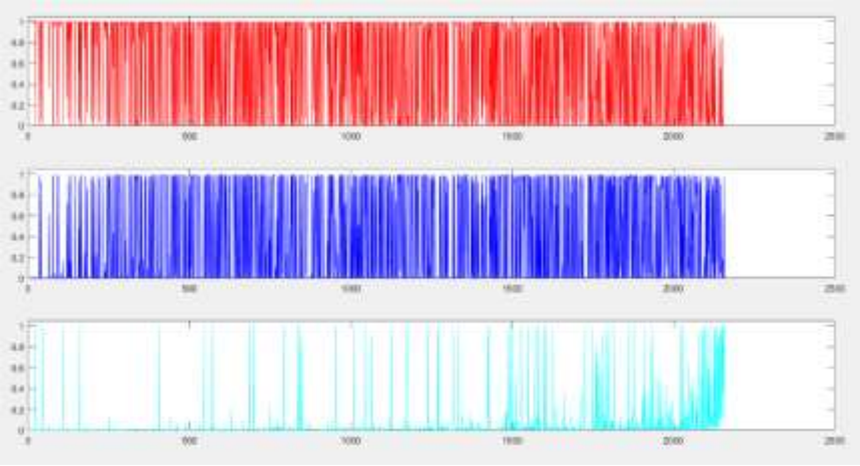
Gaussian Mixture Model

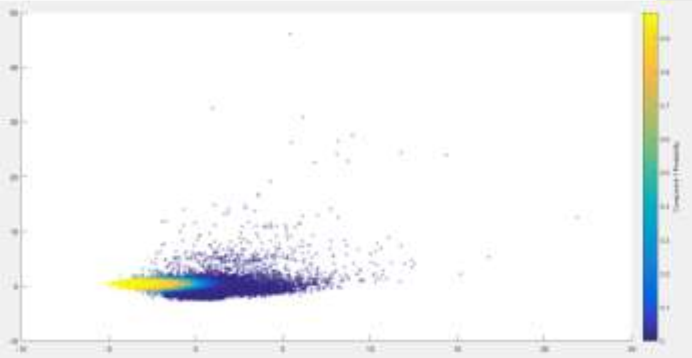
CFM	Actual–Class 1	Actual–Class 2	Actual – Class 3	Total Predicted
Predicted – Class 1	3009	5086	618	8713
Predicted– Class 2	1537	7284	3436	12257
Predicted– Class 3	24	261	358	643
Total Actual	4570	12631	4412	21613

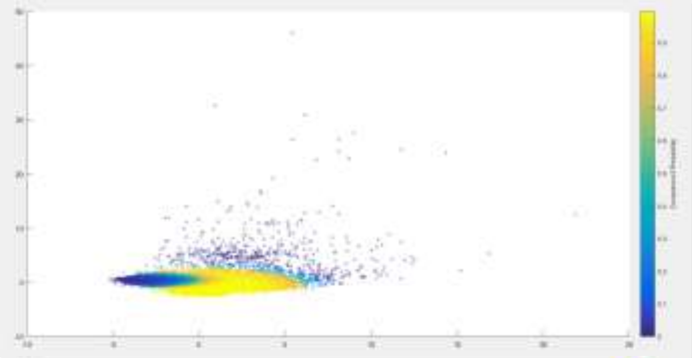
Accuracy = 0.49
Negative Log-Likelihood = 6.85e+04 = 68544

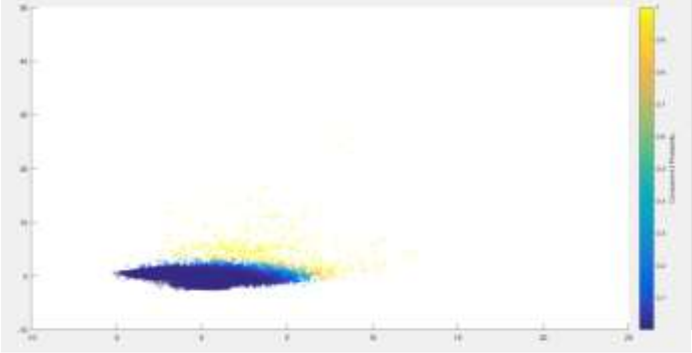
	Class 1	Class 2	Class 3
Recall	0.65	0.58	0.08
Precision	0.35	0.59	0.56
AUC	0.7176	0.5517	0.7943











Algorithm Comparisons

Regression Algorithms:

- **Linear Regression:**
  - R-squared: 0.913223143
  - Root Mean Squared Error: 0.349
- **Neural Networks:**
  - R-Squared = 0.9142
  - Root Mean Squared Error = 0.0015
  - Mean Absolute Error = 0.0015
- **Random Forest:**
  - R-squared = 0.825
  - Adjusted R-Squared = 0.825
  - Root Mean Squared Error = 0.217
  - Mean Absolute Error = 0.157

Algorithm Comparisons

Classification/Clustering Algorithms:

- **Gaussian Mixture Model:**
  - Accuracy = 0.49
  - Negative Log-Likelihood = 6.85e+04 = 68544

	Class 1	Class 2	Class 3
Recall	0.65	0.58	0.08
Precision	0.35	0.59	0.56
AUC	0.7176	0.5517	0.7943

- **Support Vector Machine:**
  - Accuracy= 0.73
  - AUC :[0.86 58,0.5426,0.20]

	Class 1	Class 2	Class 3
Precision	0.5371	0.69	0.87
Recall	0.69	0.52	0.35
AUC	0.86 58	0.5426	0.20

Q & A