**Final Project**

***Introduction:***

This project is to use big data knowledge to analysis and predict housing information in King County.

The data used for this project is Kaggle King County Housing Data and can be downloaded from: <https://www.kaggle.com/harlfoxem/housesalesprediction>

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

Below is the list of all column’s definition within this dataset:

|  |  |
| --- | --- |
| Column | Description |
| id | Unique ID for each home sold |
| date | Date of the home sale |
| price | Price of each home sold |
| bedrooms | Number of bedrooms |
| bathrooms | Number of bathrooms, where .5 accounts for a room with a toilet but no shower |
| sqft\_living | Square footage of the apartments interior living space |
| sqft\_lot | Square footage of the land space |
| floors | Number of floors |
| waterfront | A dummy variable for whether the apartment was overlooking the waterfront or not |
| view | An index from 0 to 4 of how good the view of the property was |
| condition | An index from 1 to 5 on the condition of the apartment, |
| grade | An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design. |
| sqft\_above | The square footage of the interior housing space that is above ground level |
| sqft\_basement | The square footage of the interior housing space that is below ground level |
| yr\_built | The year the house was initially built |
| yr\_renovated | The year of the house’s last renovation |
| zipcode | What zipcode area the house is in |
| lat | Lattitude |
| long | Longitude |
| sqft\_living15 | The average square footage of interior housing living space for the nearest 15 neighbors |
| sqft\_lot15 | The average square footage of the land lots of the nearest 15 neighbors |

***Analysis Practice:***

1. (Statistics) What is the average home price in the zip code 98034 and what is the standard deviation.

For this problem, we basically need first create a sub data frame, then use the sub data frame to run embedded statistics function.

Python code and result:

|  |
| --- |
| kirkland = data.loc[data["zipcode"]== 98034]  price = kirkland["price"]  p\_average = sum(price)/len(price)  p\_sd = np.std(price)  print("Average home price in 98034 is:", p\_average)  print("Zip code 98034 home price standard deviation is: ", p\_sd) |
| Average home price in 98034 is: 521652.8587155963  Zip code 98034 home price standard deviation is: 309341.42431639455 |

1. (Regression) What are the best predictors for home price from the ones in the file? Show the model.

For this problem, we first use our general knowledge to pick the columns which are most relative to home price, then create the model

Python code and result summary:

|  |
| --- |
| price = data["price"]  bedrooms = data["bedrooms"]  bathrooms = data["bathrooms"]  sqft\_living = data["sqft\_living"]  sqft\_lot = data["sqft\_lot"]  floors = data["floors"]  waterfront = data["waterfront"]  condition = data["condition"]  yr\_built = data["yr\_built"]    df = pd.DataFrame({"bedrooms" : bedrooms,                     "bathrooms" : bathrooms,  "sqft\_living" : sqft\_living,                     "sqft\_lot" : sqft\_lot,  "floors" : floors,                     "waterfront": waterfront, "condition": condition,                     "yr\_built": yr\_built                     })  Y = price  X = df  #X = sm.add\_constant(X)    model = sm.OLS(Y, X)  result = model.fit()    print(result.summary()) |
|  |

After examining the result, I see the R2 value is not high enough, so I practice add/removing more field. After trying all of them, the highest R2 value I can get is 0.880, while P value is small, so this is my best predictor model selection and the summary result:

|  |
| --- |
| df = pd.DataFrame({"bedrooms" : bedrooms, "grade": grade,                     "bathrooms" : bathrooms,  "sqft\_living" : sqft\_living,                      "sqft\_lot" : sqft\_lot, "waterfront": waterfront,                      "view": view, "condition": condition,                     "yr\_built": yr\_built, "yr\_renovated": yr\_renovated                      }) |
|  |

1. (Decision Tree) What are the best predictors for whether a home has a waterfront? Show the model.

For this problem, I still first based on general knowledge to select columns, then try to add/remove some columns to test if that column is necessary to be included into my model. My final model accuracy is at 0.9981464318813716

Python code and result:

|  |
| --- |
| import sklearn as sk  import sklearn.preprocessing as pre  import sklearn.tree as tree  from sklearn import metrics  n = len(data)  p = int(n \* 0.95)  train = data[0:p]  test = data[(p+1):n-1]  X\_train = train[[ "price", "bedrooms", "bathrooms", "sqft\_living", "sqft\_lot", "floors", "view", "condition", "grade", "sqft\_living15", "sqft\_lot15"]].values  Y\_train = train['waterfront'].values  X\_test = test[["price", "bedrooms", "bathrooms", "sqft\_living", "sqft\_lot", "floors", "view", "condition", "grade",   "sqft\_living15", "sqft\_lot15"]].values  Y\_test = test['waterfront'].values  classifier = tree.DecisionTreeClassifier(max\_depth=5)  my\_tree = classifier.fit(X\_train, Y\_train)  Y\_predict = my\_tree.predict(X\_test)  print(Y\_predict)  print("Accuracy:",metrics.accuracy\_score(Y\_test, Y\_predict)) |
| [0 0 0 ... 0 0 0]  Accuracy: 0.9981464318813716 |

Generate tree picture:

|  |
| --- |
| out = open('C:\LWTECH\CSD438-BigData\\tree3.dot', 'w')  dot\_output = tree.export\_graphviz(my\_tree, out\_file=out, feature\_names=[ "price", "bedrooms", "bathrooms", "sqft\_living", "sqft\_lot", "floors", "view",                  "condition", "grade",  "sqft\_living15", "sqft\_lot15"],                   class\_names=[ 'Waterfront', 'NotWaterfront']) |
| C:\LWTECH\CSD438-BigData>dot -Tpng tree3.dot > tree3.png |
| Diagram  Description automatically generated |
|  |

I use this model to predict a few sets data and the prediction is accurate:

|  |
| --- |
| print(my\_tree.predict([[1175000, 2, 2.5, 1770, 7155, 2, 5, 3, 8, 2410, 10476]]))  print(my\_tree.predict([[335000, 3,  2,  1410,   44866,  1, 6,   4,  7,  2950,   29152]]))  print(my\_tree.predict([[835000, 2,  2,  1410,   44866,  2, 5,   1,  2,  1000,   20000]])) |
| [1]  [0]  [1] |

If change the second set data column “price” value and third data “view” value:

|  |
| --- |
| print(my\_tree.predict([[1175000, 2, 2.5, 1770, 7155, 2, 5, 3, 8, 2410, 10476]]))  print(my\_tree.predict([[1335000,    3,  2,  1410,   44866,  1, 6,   4,  7,  2950,   29152]]))  print(my\_tree.predict([[835000, 2,  2,  1410,   44866,  2, 2,   1,  2,  1000,   20000]])) |
| [1]  [1]  [0] |

It means price and view score can significantly impact the prediction.

1. (Clustering) Cluster the data using these columns: bedrooms, bathrooms, sqft\_living, floors, waterfront, price. Name the clusters.

For this problem, I tried 3, 4, 5 clusters and compared. 3 clusters and 4 clusters are more meaningful. I pick 4 clusters as my final model:

|  |
| --- |
| df = data[['bedrooms', 'bathrooms', 'sqft\_living', 'floors', 'waterfront', 'price']]  cluster = AgglomerativeClustering(n\_clusters=4, affinity='euclidean', linkage='ward')  cluster.fit(df.values)    print(cluster.labels\_)  cluster0 = df[cluster.labels\_==0]  cluster1 = df[cluster.labels\_==1]  cluster2 = df[cluster.labels\_==2]  cluster3 = df[cluster.labels\_==3]    import statistics as st    print("Luxury Expensive Home with Great View:")  for i in range(6):      print(str(round(st.mean(cluster0.iloc[:,i]), 2)) + ": " + cluster0.columns[i])    print("\n")  print("Good Sized Family Home:")  for i in range(6):      print(str(round(st.mean(cluster1.iloc[:,i]), 2)) + ": " + cluster0.columns[i])  print("\n")  print("Basic Living Home:")  for i in range(6):      print(str(round(st.mean(cluster2.iloc[:,i]), 2)) + ": " + cluster0.columns[i])  print("\n")  print("Executive Home:")  for i in range(6):      print(str(round(st.mean(cluster3.iloc[:,i]), 2)) + ": " + cluster0.columns[i])  print("\n") |
| [2 1 2 ... 2 2 2]  Luxury Expensive Home with Great View:  4.18: bedrooms  3.43: bathrooms  4252.06: sqft\_living  1.85: floors  0.12: waterfront  1959445.51: price  Good Sized Family Home:  3.53: bedrooms  2.29: bathrooms  2308.76: sqft\_living  1.59: floors  0.0: waterfront  627535.8: price  Basic Living Home:  3.14: bedrooms  1.82: bathrooms  1635.71: sqft\_living  1.37: floors  0.0: waterfront  325844.81: price  Executive Home:  3.94: bedrooms  2.83: bathrooms  3249.66: sqft\_living  1.76: floors  0.02: waterfront  1058423.53: price |

1. (Forecasting) What is the expected average home price for January 2016 based on the average home prices from previous months?

|  |
| --- |
| avg\_all = data.groupby([data["date"].str[:6]], as\_index=False).agg(avg\_price=pd.NamedAgg(column="price", aggfunc="mean"))  avg\_all["Month"] = [1,2,3,4,5,6,7,8,9,10,11,12,13]  print( "Previous monthly average sale price list is: \n" , avg\_all)  month\_avg = avg\_all["avg\_price"]  month = avg\_all["Month"]  month = sm.add\_constant(month)  month\_model = sm.OLS(month\_avg, month)  month\_fit = month\_model.fit()  month.loc[20] = [1,21]  #2016/1  month\_new = month.loc[20:20]  predictions = month\_fit.predict(month\_new)  prediction = predictions.values  print("Predicted monthly average sale price for Jan 2016 is: ",  prediction) |
| Predicted monthly average sale price for Jan 2016 is: [536060.17773442] |

1. (Hierarchical Clustering) Use dendrogram to show cluster for zipcode and price column.

This is my pick problem to use dendrogram define clusters. Python code and plotted graph:

|  |
| --- |
| df1 = data[["zipcode","price"]]  import scipy.cluster.hierarchy as shc  plt.figure(figsize=(10, 7))  plt.title("Price Dendograms")  dend = shc.dendrogram(shc.linkage(df1, method='ward'))  plt.show() |

|  |
| --- |
|  |

1. (Cluster plot) Use dendrogram defined number of clusters to plot the clusters .

|  |
| --- |
| cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')  cluster.fit\_predict(df1)  plt.figure(figsize=(10, 7))  plt.scatter(df1.iloc[:,0:1], df1.iloc[:,1], c=cluster.labels\_, cmap='rainbow')  plt.show() |
|  |

***Conclusion:***

1. During May 2014 ~ May 2015, average price of sold homes in 98034 is $521,652.
2. These home features have direct impact on home price: number of bedrooms, number of bathrooms, size of living space, size of home lot, number of floors, home face a water or not, the home conditions, and how old of this home.
3. When a home has high price and high score of view, the high probability is that this home is a waterfront home.
4. Over a million dollars home usually have 4+ bedrooms, and if home price reach around 2 million dollars, the home more likely face lake/ocean. A normal family home usually not close to water.
5. November, December, January and February home sold at lower average price around $500s~$520s. Jan.2015 sold home average price at $525,871, while in 2016 January home sold t average $536,060, which mean the home price increased 2% in a year.
6. It is a good practice to use dendrogram to plot cluster links first then decide how many clusters is appropriate for a dataset.
7. Based on zip code area cluster chart, my conclusion is that between May 2014 and May 2015, most expensive homes are sold in east side, while in Seattle city area, most homes are normal family home.