**CAPSTONE PROJECT1: Real Estate Case Study**

**PROBLEM STATEMENT**

A banking institution requires actionable insights from the perspective of Mortgage-Backed Securities, Geographic Business Investment and Real Estate Analysis. The objective is to identify white spaces/potential business in the mortgage loan. The mortgage bank would like to identify potential monthly mortgage expenses for each of region based on factors which are primarily monthly family income in a region and rented value of the real estate. Some of the regions are growing rapidly and Competitor banks are selling mortgage loans to subprime customers at a lower interest rate. The bank is strategizing for better market penetration and targeting new customers. A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. This would help to monitor the key metrics and trends. The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental income, monthly mortgage and owner’s cost, family income vs mortgage cost comparison across different regions. The metrics are described not to limit the dashboard to these few only.

**1. Import data**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** itertools **import** cycle

​

pd.set\_option('max\_columns', 90)

pd.set\_option('max\_rows', 90)

plt.style.use('bmh')

color\_pal **=** plt.rcParams['axes.prop\_cycle'].by\_key()['color']

color\_cycle **=** cycle(plt.rcParams['axes.prop\_cycle'].by\_key()['color'])

train **=** pd.read\_csv('train.csv')

test **=** pd.read\_csv('test.csv')

train.head()

test.head()

**2. Figure out the primary key and look for the requirement of indexing**

**UID is the primary Key**

**3. Gauge the fill rate of the variables and devise plans for missing value treatment. Please explain explicitly the reason for the treatment chosen for each variable.**

train.columns

Out[5]:

train.dtypes

train.columns[:5]

train[train.columns[0:20]].head()

cat\_columns **=** ['UID', 'COUNTYID', 'STATEID', 'state', 'state\_ab', 'city', 'place', 'type', 'primary', 'zip\_code', 'area\_code']

train[cat\_columns].dtypes

train.shape

train.columns

test.columns

*# UID is unique userID value in the train and test dataset. So an index can be created from the UID feature*

train.set\_index(keys**=**['UID'],inplace**=True**)*#Set the DataFrame index using existing columns.*

test.set\_index(keys**=**['UID'],inplace**=True**)

*# Handling Missing value*

train.isnull().sum()**/**len(train)**\***100

train**=**train.drop(['BLOCKID','SUMLEVEL'],axis**=**1)

test.isnull().sum()**/**len(test)**\***100

test**=**test.drop(['BLOCKID','SUMLEVEL'],axis**=**1)

*# Imputing missing values with mean*

missing\_train\_cols**=**[]

**for** col **in** train.columns:

**if** train[col].isna().sum() **!=**0:

missing\_train\_cols.append(col)

print(missing\_train\_cols)

missing\_test\_cols**=**[]

**for** col **in** test.columns:

**if** test[col].isna().sum() **!=**0:

missing\_test\_cols.append(col)

print(missing\_test\_cols)

*# Missing cols are all numerical variables*

**for** col **in** train.columns:

**if** col **in** (missing\_train\_cols):

train[col].replace(np.nan,train[col].mean(),inplace**=True**)

**for** col **in** test.columns:

**if** col **in** (missing\_test\_cols):

test[col].replace(np.nan,test[col].mean(),inplace**=True**)

train.isna().sum().sum()

test.isna().sum().sum()

**Week 1 Exploratory Data Analysis**

df **=** train[train['pct\_own']**>**0.1]

df.shape

Out[26]:

(26565, 77)

df **=** df.sort\_values(by**=**'second\_mortgage',ascending**=False**)

pd.set\_option('display.max\_columns', **None**)

df.head()

**import** plotly.express **as** px

**import** plotly.graph\_objects **as** go

*# Visualization 1 (Geo-Map):*

fig **=** go.Figure(data**=**go.Scattergeo(

lat **=** top\_2500\_second\_mortgage\_pctown\_10['lat'],

lon **=** top\_2500\_second\_mortgage\_pctown\_10['lng']),

)

fig.update\_layout(

geo**=**dict(

scope **=** 'north america',

showland **=** **True**,

landcolor **=** "rgb(212, 212, 212)",

subunitcolor **=** "rgb(255, 255, 255)",

countrycolor **=** "rgb(255, 255, 255)",

showlakes **=** **True**,

lakecolor **=** "rgb(255, 255, 255)",

showsubunits **=** **True**,

showcountries **=** **True**,

resolution **=** 50,

projection **=** dict(

type **=** 'conic conformal',

rotation\_lon **=** **-**100

),

lonaxis **=** dict(

showgrid **=** **True**,

gridwidth **=** 0.5,

range**=** [ **-**140.0, **-**55.0 ],

dtick **=** 5

),

lataxis **=** dict (

showgrid **=** **True**,

gridwidth **=** 0.5,

range**=** [ 20.0, 60.0 ],

dtick **=** 5

)

),

title**=**'Top 2,500 locations with second mortgage is the highest and percent ownership is above 10 percent')

fig.show()

train['bad\_debt']**=**train['second\_mortgage']**+**train['home\_equity']**-**train['home\_equity\_second\_mortgage']

*# Visualization 2:*

train['bins\_bad\_debt'] **=** pd.cut(train['bad\_debt'],bins**=**[0,0.1,.5,1], labels**=**["less than 10%","10-50%","50-100%"])

train.groupby(['bins\_bad\_debt']).size().plot(kind**=**'pie',subplots**=True**,startangle**=**90, autopct**=**'%1.1f%%')

plt.title('Bad Debt pct')

plt.ylabel("")

​

plt.show()

*# Visualization 3:*

train['bins\_debt'] **=** pd.cut(train['debt'],bins**=**[0,0.1,.5,1], labels**=**["less than 10%","10-50%","50-100%"])

train.groupby(['bins\_debt']).size().plot(kind**=**'pie',subplots**=True**,startangle**=**90, autopct**=**'%1.1f%%')

plt.title('Debt pct')

plt.ylabel("")

​

plt.show()

cols**=**['second\_mortgage','home\_equity','debt','bad\_debt']

df\_box\_hamilton**=**train.loc[train['city'] **==** 'Hamilton']

df\_box\_manhattan**=**train.loc[train['city'] **==** 'Manhattan']

df\_box\_city**=**pd.concat([df\_box\_hamilton,df\_box\_manhattan])

df\_box\_city.head(4)

*# Visualization 4:*

plt.figure(figsize**=**(10,5))

sns.boxplot(data**=**df\_box\_city,x**=**'second\_mortgage', y**=**'city',width**=**0.5,palette**=**"Set3")

plt.show()

*# Visualization 5:*

plt.figure(figsize**=**(10,5))

sns.boxplot(data**=**df\_box\_city,x**=**'home\_equity', y**=**'city',width**=**0.5,palette**=**"Set3")

plt.show()

*# Visualization 6:*

plt.figure(figsize**=**(10,5))

sns.boxplot(data**=**df\_box\_city,x**=**'debt', y**=**'city',width**=**0.5,palette**=**"Set3")

plt.show()

*# Visualization 7:*

plt.figure(figsize**=**(10,5))

sns.boxplot(data**=**df\_box\_city,x**=**'bad\_debt', y**=**'city',width**=**0.5,palette**=**"Set3")

plt.show()

*# Visualization 8:*

sns.distplot(train['hi\_mean'])

plt.title('Household income distribution chart')

plt.show()

*# Visualization 9:*

sns.distplot(train['family\_mean'])

plt.title('Family income distribution chart')

plt.show()

*# Visualization 10:*

sns.distplot(train['family\_mean']**-**train['hi\_mean'])

plt.title('Remaining income distribution chart')

plt.show()

*# Visualization 11:*

sns.distplot(train['pop'])

plt.title('Population distribution chart')

plt.show()

*# Visualization 12:*

sns.distplot(train['male\_pop'])

plt.title('Male population distribution chart')

plt.show()

*# Visualization 13:*

sns.distplot(train['female\_pop'])

plt.title('Female population distribution chart')

plt.show()

*# Visualization 14:*

sns.distplot(train['male\_age\_median'])

plt.title('Male age distribution chart')

plt.show()

*# Visualization 15:*

sns.distplot(train['female\_age\_median'])

plt.title('Female age distribution chart')

plt.show()

train["pop\_density"]**=**train["pop"]**/**train["ALand"]

test["pop\_density"]**=**test["pop"]**/**test["ALand"]

*# Visualization 16:*

sns.distplot(train['pop\_density'])

plt.title('Population density distribution chart')

plt.show()

*# Visualization 17:*

sns.boxplot(train['pop\_density'])

plt.title('Population density distribution chart')

plt.show()

train["median\_age"]**=**(train["male\_age\_median"]**+**train["female\_age\_median"])**/**2

test["median\_age"]**=**(test["male\_age\_median"]**+**test["female\_age\_median"])**/**2

train[['male\_age\_median','female\_age\_median','male\_pop','female\_pop','median\_age']].head()

*# Visualization 18:*

sns.distplot(train['median\_age'])

plt.title('Age median distribution chart')

plt.show()

train["pop"].describe()

train['pop\_bins']**=**pd.cut(train['pop'],bins**=**5,labels**=**['very low','low','medium','high','very high'])

train[['pop','pop\_bins']]

train['pop\_bins'].value\_counts()

train.groupby(by**=**'pop\_bins')[['married','separated','divorced']].count()

train.groupby(by**=**'pop\_bins')[['married','separated','divorced']].agg(["mean", "median"])

*# Visualization 19:*

pop\_bin\_married**=**train.groupby(by**=**'pop\_bins')[['married','separated','divorced']].agg(["mean"])

sns.lineplot(data**=**pop\_bin\_married)

plt.show()

rent\_state\_mean**=**train.groupby(by**=**'state')['rent\_mean'].agg(["mean"])

rent\_state\_mean.head()

income\_state\_mean**=**train.groupby(by**=**'state')['family\_mean'].agg(["mean"])

income\_state\_mean.head()

rent\_perc\_of\_income**=**rent\_state\_mean['mean']**/**income\_state\_mean['mean']

rent\_perc\_of\_income.head(10)

*#overall level rent as a percentage of income*

sum(train['rent\_mean'])**/**sum(train['family\_mean'])

*#Correlation analysis and heatmap*

train[["COUNTYID","STATEID","zip\_code", "type","pop","family\_mean",'second\_mortgage', 'home\_equity', 'debt','hs\_degree','median\_age','pct\_own', 'married','separated','divorced']]

sns.heatmap(train[["COUNTYID","STATEID","zip\_code", "type","pop","family\_mean",'second\_mortgage', 'home\_equity', 'debt','hs\_degree','median\_age','pct\_own', 'married','separated', 'divorced']].corr())

**Data Pre-processing:**

The economic multivariate data has a significant number of measured variables. The goal is to find where the measured variables depend on a number of smaller unobserved common factors or latent variables. 2. Each variable is assumed to be dependent upon a linear combination of the common factors, and the coefficients are known as loadings. Each measured variable also includes a component due to independent random variability, known as “specific variance” because it is specific to one variable. Obtain the common factors and then plot the loadings. Use factor analysis to find latent variables in our dataset and gain insight into the linear relationships in the data. Following are the list of latent variables:

• Highschool graduation rates

• Median population age

• Second mortgage statistics

• Percent own

• Bad debt expense

**from** sklearn.decomposition **import** FactorAnalysis

fa **=** FactorAnalysis(n\_components**=**5,random\_state**=**11)

train\_transformed **=** fa.fit\_transform(train.select\_dtypes(exclude**=**('object','category')))

train\_transformed.shape

train\_transformed

Out[79]:

array([[ 0.05697554, -0.05009055, 1.24976488, -0.05628299, 0.72025225],

[-0.10031282, 0.01329662, 0.1106829 , -1.10456586, -0.31552827],

[-0.04700012, -0.00998672, 0.13037883, 0.81365251, 0.90142656],

...,

[ 0.93590311, -0.36941991, -0.96872656, 0.11395017, -0.89856758],

[-0.08691309, 0.00750638, -0.88775357, 3.86619218, 1.59161861],

[-0.0954242 , 0.01057289, -1.33120384, -0.71237903, -0.07688444]])

x\_train **=** pd.read\_csv('train.csv')

x\_test **=** pd.read\_csv('test.csv')

x\_train.drop(['BLOCKID','SUMLEVEL'],axis**=**1,inplace**=True**)

x\_train.dropna(axis**=**0,inplace**=True**)

x\_train.head()

x\_train.drop\_duplicates(inplace**=True**)

x\_train.shape

Out[84]:

(26585, 78)

x\_test.head()

x\_test.shape

Out[86]:

(11709, 80)

x\_test.drop(['BLOCKID','SUMLEVEL'],axis**=**1,inplace**=True**)

x\_test.isna().sum()

x\_test.dropna(axis**=**0,inplace**=True**)

x\_test.drop\_duplicates(inplace**=True**)

x\_test.shape

x\_test.isna().sum()

x\_train.dtypes

imp\_feature **=** x\_train.select\_dtypes(exclude**=**('object','category'))

imp\_feature

imp\_feature.shape

to\_drop **=** ['UID','COUNTYID', 'STATEID', 'zip\_code', 'area\_code', 'lat', 'lng']

**for** col **in** imp\_feature.columns:

**if** col **in** to\_drop:

imp\_feature.drop(col,axis**=**1,inplace**=True**)

imp\_feature.head()

x\_train\_features **=** imp\_feature[['pop','rent\_median','hi\_median','family\_median','hc\_mean','second\_mortgage','home\_equity','debt','hs\_degree','pct\_own','married','separated','divorced']]

x\_train\_features.head()

x\_train\_features.shape

Out[102]:

(26585, 13)

y\_train **=** imp\_feature['hc\_mortgage\_mean']

x\_test\_feature **=** x\_test[['pop','rent\_median','hi\_median','family\_median','hc\_mean','second\_mortgage','home\_equity','debt','hs\_degree','pct\_own','married','separated','divorced']]

**from** sklearn.linear\_model **import** LinearRegression

le **=** LinearRegression()

le.fit(x\_train\_features,y\_train)

y\_pred **=** le.predict(x\_test\_feature)

y\_test **=** x\_test['hc\_mortgage\_mean']

**from** sklearn.metrics **import** r2\_score,mean\_squared\_error

r2\_score(y\_test,y\_pred)

np.sqrt(mean\_squared\_error(y\_test,y\_pred))

*# Visualization 21:*

sns.distplot(y\_pred)

plt.show()