MID TERM REPORT

TEAM 7: Vipra Shah , Snigdha Joshi

Part1

Read summarized origination data

```
In [10]: import pandas as pd
         import os, matplotlib
         import matplotlib.pyplot as plt
         import numpy as np
         import seaborn as sns
         %matplotlib inline

         sample_clean_file = os.getcwd() + "/sample_orig_combined.csv"
         sample_df = pd.read_csv(sample_clean_file,low_memory=False)
         sample_df.head(2)
```

Out[10]:

	credit_score	first_payment_date	fthb_flag	matr_date	msa	mortage_insurance_pct	no_of_units	occupancy_status	cltv	dti_ratio	...	p
0	591	200504	N	203503	39100	0.0	1	O	48	34	...	N
1	792	200503	N	203502	39100	0.0	1	O	90	33	...	N

2 rows × 26 columns

Read summarized performance data

```
In [11]: performance_file = os.getcwd() + "/Summarized_performance_data.csv"
         perf_df = pd.read_csv(performance_file, low_memory=False)
         perf_df.head(2)
```

Out[11]:

	loan_seq_number	month	max_current_actual_upb	min_current_actual_upb	delq_status	loan_age	rem_months	repurchase_flag	modi
0	F105Q1000064	200912	62000.0	0.0	0	57	303	NaN	Not M
1	F105Q1000076	201011	197000.0	0.0	0	69	291	NaN	Not M

2 rows × 24 columns

Add a new column for Year and Read merged file

```
In [12]: perf_df['Year'] = ['20'+ x for x in (perf_df['loan_seq_number'].apply(lambda x: x[2:4]))]

In [13]: perf_df['Year'].unique()

Out[13]: array(['2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012',
                '2013', '2014', '2015', '2016'], dtype=object)

In [14]: merged_df = pd.merge(sample_df,perf_df,on="loan_seq_number",how="right")

In [15]: merged_df.head(2)
```

Out[15]:

	credit_score	first_payment_date	fthb_flag	matr_date	msa	mortage_insurance_pct	no_of_units	occupancy_status	cltv	dti_ratio	...
0	591.0	200504.0	N	203503.0	39100.0	0.0	1.0	O	48.0	34.0	...
1	792.0	200503.0	N	203502.0	39100.0	0.0	1.0	O	90.0	33.0	...

2 rows × 50 columns

```
In [16]: merged_df.shape

Out[16]: (574957, 50)
```

Group merged file based on year

In [17]:

```python
yearwise_df = pd.DataFrame()
grouped = merged_df.groupby('Year')
yearwise_df = yearwise_df.append(grouped.aggregate(np.mean))
yearwise_df.drop(['first_payment_date', 'matr_date', 'msa','zipcode', 'ddlpi','month', 'zero_bal_date', 'rem_months'], axis=
#yearwise_df = yearwise_df.transpose()
yearwise_df.head(2)
```

Out[17]:

	credit_score	mortage_insurance_pct	no_of_units	cltv	dti_ratio	original_upb	original_ltv	original_int_rt	original_loan_tern
Year									
2005	724.727288	3.027869	1.022480	70.694997	34.262401	177046.460124	68.955302	5.796721	324.040912
2006	723.229643	3.226935	1.024275	72.977019	35.954134	186530.265996	70.394848	6.397613	337.934244

2 rows × 26 columns

UPB trends over the years

In [18]:

```python
def upb_trends_over_time():
    original_upb = yearwise_df['original_upb']
    max_current_actual_upb = yearwise_df['max_current_actual_upb']
    min_current_actual_upb = yearwise_df['min_current_actual_upb']
    year = perf_df['Year'].drop_duplicates()
    #year_df = pd.DataFrame(perf_df['Year'].drop_duplicates()).reset_index()
    #year_df.columns = ['Count', 'Year']
    #year_df = year_df.ix[(year_df['Year'] == '2007') | (year_df['Year'] == '2008') | (year_df['Year'] == '2009')]
    plt.figure(num=None, figsize=(14, 12),dpi=50, facecolor='c', edgecolor='b')
    ax1=plt.subplot(211)
    plt.plot(year,original_upb,'y--',year,max_current_actual_upb,'b--',year,min_current_actual_upb,'r--')
    plt.xlabel('YEAR')
    plt.ylabel('UPB')
    plt.legend(['Original UPB','Max UPB','Min UPB'])
    plt.grid(True)
    plt.title('UPB Trends Over Years')

upb_trends_over_time()
```

Zero balance code trends over time

Zero Balance Code Trends over Time

In [325]:
```
total_records = merged_df.shape[0]
print("Total records is {}".format(total_records))
```

```
Total records is 574957
```

In [326]:
```
total_default_records = merged_df.ix[(merged_df['zero_balance_code'] == 3) | (merged_df['zero_balance_code'] == 6) |
                                     (merged_df['zero_balance_code'] == 9)]
count = total_default_records.shape[0]
print("Total number of default record is: {}".format(count))
default_ratio = str(round(count/total_records,2))
print("Total default ratio is {}".format(default_ratio))
```

```
Total number of default record is: 15042
Total default ratio is 0.03
```

In [327]:
```
total_prepaid_records = merged_df.ix[merged_df['zero_balance_code'] == 1]
count_prepaid = total_prepaid_records.shape[0]
print("Total number of prepaid record is {}".format(count_prepaid))
prepaid_ratio = str(round(count_prepaid/total_records,2))
print("Total prepaid ratio is {}".format(prepaid_ratio))
```

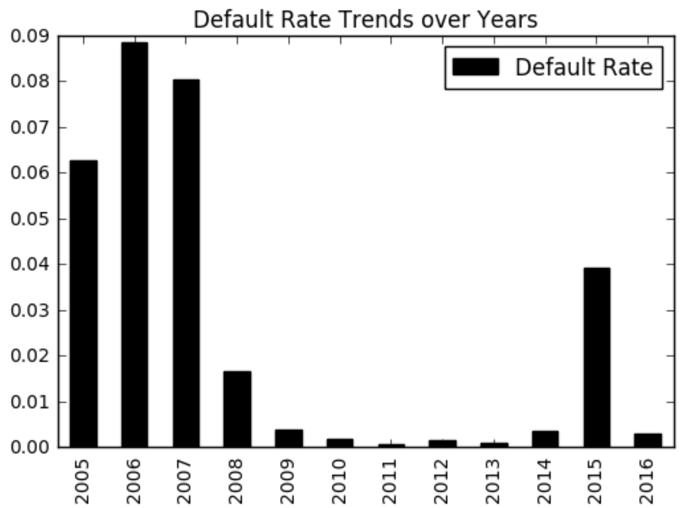
```
Total number of prepaid record is 314001
Total prepaid ratio is 0.55
```

In [328]:
```
total_default_records_by_year = total_default_records.groupby(perf_df['Year'])['loan_seq_number'].count()
total_default_records_by_year
```

Plot graph for Default Rate Trends over Years

In [331]:
```
default_rate_by_year = pd.concat([total_default_records_by_year,total_records_by_year],axis = 1, join='ou
default_rate_by_year['Default Rate'] = default_rate_by_year['Default Records Number']/default_rate_by_yea
default_rate_by_year.plot(title ="Default Rate Trends over Years",y='Default Rate', color='k', kind='bar'
```

Out[331]: <matplotlib.axes._subplots.AxesSubplot at 0x22553ce82b0>



Part2

Prediction:

1. Building regression model:

```
---------------Linear Regression---------------
Train Data:
Mean Absolute Error:  0.215635923889
Root of Mean Squared Deviation:  0.28904986796221477
Mean Absolute Percentage Error:  3.8205931686665737
None
Test Data:
Mean Absolute Error:  0.248198869421
Root of Mean Squared Deviation:  0.322479782238875
Mean Absolute Percentage Error:  4.241883897814012
None
---------------Random Forest---------------
Train Data:
Mean Absolute Error:  0.202088167802
Root of Mean Squared Deviation:  0.26988543697197087
Mean Absolute Percentage Error:  3.5827581077356547
None
Test Data:
Mean Absolute Error:  0.231182052829
Root of Mean Squared Deviation:  0.30108961650858196
Mean Absolute Percentage Error:  3.9872682618105753
None
```

```
---------------Neural Network---------------
Train Data
MSE, epoch 2: 0.06650558148048123
RMSE, epoch 2: 0.25788676096395724
```

```
Test Data:
MSE:  0.130481074978
RMSE:  0.36122164245478
```

From above 3 algorithms we chose **random forest** because:

- Better results than Linear Regression
- Lot less processing time than Neural networks(Fast and scalable)

Furthermore,

- Processing time does not increase substantially with increase in number of observations.
- Easy to interpret, adjust(tune) parameters to achieve desired results.
- It is Non-parametric, we don't have to worry about outliers.

2. **Financial crisis** (https://www.stlouisfed.org/financial-crisis/full-timeline)

During financial crisis of 2007 Freddie Mac declared following actions:

- In Q1 & 2: it will no longer buy the riskiest subprime mortgages and mortgage-related securities.
- In Q3: Countrywide Financial Corporation warns of “difficult conditions.
- In Q4: Financial market pressures intensify, reflected in diminished liquidity in interbank funding markets.

Train	Q12007	Q22007	Q32007	Q42007
MAE	0.2294	0.2387	0.236	0.2574
RMS	0.3023	0.3134	0.3133	0.3409
MAPE	3.7052	3.797	3.5654	4.0535

Test	Q22007	Q32007	Q42007	Q12008
MAE	0.2947	0.2889	0.2721	0.3013
RMS	0.3723	0.3689	0.3564	0.3866
MAPE	4.8494	4.6764	4.1559	4.8085

MAE	28.4656	21.03058	15.29661	17.05517
RMS	23.1558	17.709	13.75678	13.40569
MAPE	30.8809	23.16039	16.56196	18.62588

As we can see, difference between Training and Testing MAE and RMS has increasing substantially in Q1 and Q2 of year 2007.

1. In Q12007 train and test data RMS has increased by 23.15%
2. As model is trained, RMS is decreasing

The federal takeover of Freddie Mac in **September 2008** improved situations. It was one of the financial events among many in the ongoing subprime mortgage crisis.

What the Fed did:

The Fed initiated purchases of $500 billion in mortgage-backed securities.

- It announced purchases of up to $100 billion in debt obligations of mortgage giants Fannie Mae, Freddie Mac, Ginnie Mae and Federal Home Loan Banks.
- The Fed cut the key interest rate to near zero, Dec. 16, 2008.
- In March 2009, the Fed expanded the mortgage buying program and said it would purchase $750 billion more in mortgage-backed securities.
- The Fed also announced it would invest another $100 billion in Freddie debt and purchase up to $300 billion of longer-term Treasury securities over a period of six months.
- In the first quarter of 2010, with a total of $1.25 trillion in purchases of mortgage-backed securities and $175 billion of agency debt purchases.

As a result:

Mortgage rates dropped significantly, to as low as 5%, in year 2009.

Train	Q12009	Q22009	Q32009	Q42009
MAE	0.2261	0.2051	0.2339	0.176
RMS	0.3031	0.2753	0.2995	0.2257
MAPE	4.5276	4.1961	4.5929	3.5638

Test	Q22009	Q32009	Q42009	Q12010
MAE	0.2304	0.2121	0.2662	0.187
RMS	0.3172	0.2917	0.3358	0.2431
MAPE	4.5705	4.2903	5.1118	3.7502

MAE	1.901813357	3.412969283	13.80932022	6.25
RMS	4.651930056	5.957137668	12.12020033	7.709348693
MAPE	0.947521866	2.244941732	11.2978728	5.230372075

As we can see, in Q1 & Q2 2007 RMS and MAE is substantially lower than 2007.

As situations began to improve in 2009, observations recorded changes, which in turn increased prediction error.

Economic Boom

The 1990s economic boom in the United States was an extended period of economic prosperity, during which GDP increased continuously for almost ten years (the longest recorded expansion in the history of the United States). It commenced after the end of the early 1990s recession in March 1991, and ended in March 2001 with the start of the early 2000s recession, following the bursting of the dot com bubble**.**

1995-2000 is also remembered for a series of global economic financial crises that threatened the U.S. economy.

Despite occasional stock market downturns and some distortions in the trade deficit, the US economy remained resilient until the dot-com bubble peaked in March 2000.

Train	Q11999	Q21999	Q31999	Q41999
MAE	0.2107	0.2347	0.259	0.2358
RMS	0.292	0.3156	0.3564	0.3305
MAPE	3.0332	3.28	3.3647	2.9768

Test	Q21999	Q31999	Q41999	Q12000
MAE	0.3438	0.2722	0.2773	0.2645
RMS	0.4246	0.3686	0.3818	0.3663
MAPE	5.0457	3.8271	3.5847	3.3405

MAE	63.17038443	15.97784406	7.065637066	12.17133164
RMS	45.4109589	16.79340938	7.126823793	10.83207262
MAPE	66.34907029	16.67987805	6.538472969	12.21781779

Observations:

1. As financial crisis began towards 2000, there were drastic changes in values in datasets.
2. Which caused RMS to be increased drastically in Q12009.

3. Model started to get settled towards Q2 and Q3 of 2009.

The Fed was buying another $40 billion in mortgage-backed investments each month until the economy improved. That's on top of the tens of billions of dollars in mortgages it already had been buying each month, making U.S. banks flush with cash.

Starting in Dec 2013, the Fed began to reduce its $85 billion-per-month asset purchases by $10 billion per month at each Fed meeting, cutting them to $35 billion in June 2014.

Train	Q12013	Q22013	Q32013	Q42013
MAE	0.1599	0.1765	0.2508	0.1744
RMS	0.2094	0.2299	0.3233	0.2269
MAPE	4.7259	5.0445	6.2844	4.0814

Test	Q22013	Q32013	Q42013	Q12014
MAE	0.1685	0.1868	0.2767	0.2001
RMS	0.226	0.251	0.3508	0.2603
MAPE	4.9119	5.2668	6.7444	4.5782

MAE	5.378361476	5.835694051	10.32695375	14.73623853
RMS	7.927411652	9.177903436	8.50603155	14.72014103
MAPE	3.935758268	4.406779661	7.319712303	12.17229382

Observations:

1. At the end of 2013, there was sudden increse in mortgage rates, which lasted till end of Q12014.
2. As a result, we can seen gradual increase in RMS values towards end of year.

Would you recommend using this model for the next quarter? Justify

As per observations, our model is correctly picking up economic changes.

In stable period RMS percentage differences between RMS values are rarely greater than **13%**

Therefore I would recommend to use this model for next quarters

Classification:

Download quarterly data based on User input

Load training and test quarter data into dataframe

Clean data for training and test quarter

Step1: Download the quarterly data based on the user input.

```
def downloadhistoricaldata(trainQ, testQ, t,s, flag):
    for l in t:
        if(trainQ in l['href'] or testQ in l['href']):
            c = 'https://freddiemac.embs.com/FLoan/Data/' + l['href']
            r = s.get(c)
            z = ZipFile(BytesIO(r.content))
            z.extractall(os.getcwd()+ '/data_part2')
            flag = 1
    return flag

def login(login, password, trainQ, testQ):
    flag = 0
    s = requests.Session()
    url = "https://freddiemac.embs.com/FLoan/secure/auth.php"
    url2 = "https://freddiemac.embs.com/FLoan/Data/download.php"
    browser = ms.Browser(session = s)
    print("Logging in....")
    login_page = browser.get(url)
    login_form = login_page.soup.find("form",{"class":"form"})
    login_form.find("input", {"name":"username"})["value"] = login
    login_form.find("input", {"name":"password"})["value"] = password
    response = browser.submit(login_form, login_page.url)
    login_page2 = browser.get(url2)
    print("To the continue page...")

    next_form = login_page2.soup.find("form",{"class":"fmform"})
    a= next_form.find("input",{"name": "accept"}).attrs
    a['checked']=True
```

References:

https://www.thebalance.com/stock-market-returns-by-year-2388543

http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

http://bigdata-madesimple.com/how-to-run-linear-regression-in-python-scikit-learn/

https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/

http://www.bankrate.com/finance/federal-reserve/financial-crisis-timeline.aspx

https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms