NOMURA

Mortgage-Backed Securities: Machines Learning Models on Prepayment Projections

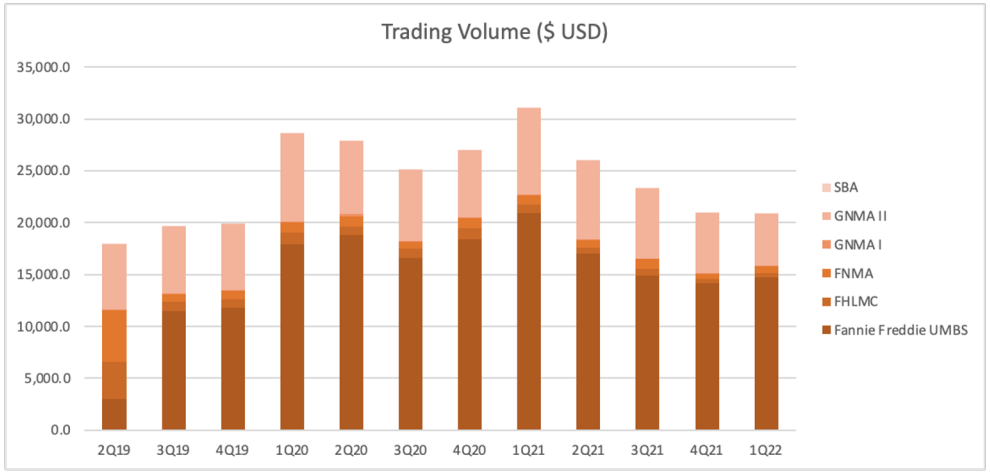
May 12th | Spring 2022

U.S. Agency Mortgage-Backed Securities market is one of the largest and most liquid bond markets worldwide

Snapshot

Mortgage-backed security is a securitized claim to the principal and interest payments generated by a pool of fixed-rate mortgages

- Issuance (as of April) $990.2 billion, -43.7% Y/Y
- Agency Trading (as of April) $271,819.3 million, -14.3% Y/Y
- Non-Agency Trading (as of April) $1,444.2 million, -9.5% Y/Y
- Outstanding (as of 4Q21) $12,201.6 billion, +9% Y/Y



Influence of Prepayment

The risk that a mortgage will be repaid earlier than the originally agreed termination date

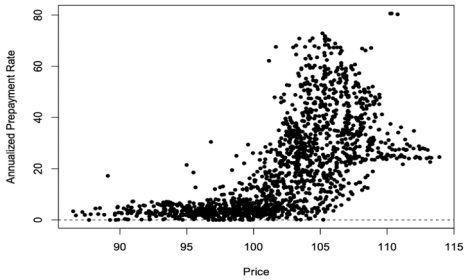
- Possibility of a prepayment risk premium; requires that prepayment dynamics be the same under risk-neutral and actual measures
- Yield spread adjustment to discount rates, known as the option-adjusted spread

Reasons to Prepay

External

- Interest Rate
- House Price
- Employment Rate
- Seasonality

Refinance



Source: SIFMA, Fannie Mae, Freddie Mac, Internal Analysis

Data Cleaning

Selection Criteria

- Amortization Type: Fixed Rate Mortgage
- Property Type: Single Family
- Loan Term: equals to 360 terms
- Loan Age: smaller than 120 terms

After selection, the total size of data becomes 17.1GB

Source: Fannie Mae, Freddie Mac, Internal Analysis

Data Cleaning

Clean Empty Columns

FMAE Empty Columns:

'Filler', 'Filler.1', 'Filler.2', 'Government Insured Guarantee', 'Modification Program', 'Modification Type', 'Number of Modifications', 'Total Capitalized Amount', 'Interest Bearing Mortgage Loan Amount', 'Original Deferred Amount', 'Current Deferred UPB', 'Loan Age As Of Modification', 'Estimated Loan-To-Value (ELTV)', 'Updated Credit Score', 'Filler.3', 'Interest Rate Step Indicator', 'Initial Step Fixed-Rate Period', 'Total Number of Steps', 'Number of Remaining Steps', 'Next Step Rate', 'Terminal Step Rate', 'Terminal Step Date', 'Step Rate Adjustment Frequency', 'Next Step Rate Adjustment Date', 'Months to Next Step Rate Adjustment Date', 'Periodic Step Cap Up Percent', 'Origination Mortgage Loan Amount', 'Origination Interest Rate', 'Origination Amortization Type', 'Origination Interest Only Loan Indicator', 'Origination First Payment Date', 'Origination Maturity Date', 'Origination Loan Term', 'Origination Loan-To-Value (LTV)', 'Origination Combined Loan-To-Value (CLTV)', 'Origination Debt-To-Income Ratio', 'Origination Credit Score', 'Filler.4', 'Filler.5', 'Filler.6', 'Origination Loan Purpose', 'Origination Occupancy Status', 'Origination Channel', 'Loan Performance History'

FMAC Empty Columns:

'Filler', 'Filler.1', 'Filler.2', 'Interest Only First Principal and Interest Payment Date', 'Months to Amortization', 'Prepayment Penalty Total Term', 'Index', 'Mortgage Margin', 'MBS PC Margin', 'Interest Rate Adjustment Frequency', 'Interest Rate Lookback', 'Interest Rate Rounding Method', 'Interest Rate Rounding Method Percent', 'Convertibility Indicator', 'Initial Fixed Rate Period', 'Next Interest Rate Adjustment Date', 'Months to Next Interest Rate Adjustment Date', 'Life Ceiling Interest Rate', 'Life Ceiling Net Interest Rate', 'Life Floor Interest Rate', 'Life Floor Net Interest Rate', 'Initial Interest Rate Cap Up Percent', 'Initial Interest Rate Cap Down Percent', 'Periodic Interest Rate Cap Up Percent', 'Periodic Interest Rate Cap Down Percent', 'Modification Program', 'Modification Type', 'Number of Modifications', 'Total Capitalized Amount', 'Interest Bearing Mortgage Loan Amount', 'Original Deferred Amount', 'Current Deferred UPB', 'Loan Age As Of Modification', 'Estimated Loan-To-Value (ELTV)', 'Updated Credit Score', 'Filler.3', 'Interest Rate Step Indicator', 'Initial Step Fixed-Rate Period', 'Total Number of Steps', 'Number of Remaining Steps', 'Next Step Rate', 'Terminal Step Rate', 'Terminal Step Date', 'Step Rate Adjustment Frequency', 'Next Step Rate Adjustment Date', 'Months to Next Step Rate Adjustment Date', 'Periodic Step Cap Up Percent', 'Origination Mortgage Loan Amount', 'Origination Interest Rate', 'Origination Amortization Type', 'Origination Interest Only Loan Indicator', 'Origination First Payment Date', 'Origination Maturity Date', 'Origination Loan Term', 'Origination Loan-To-Value (LTV)', 'Origination Combined Loan-To-Value (CLTV)', 'Origination Debt-To-Income Ratio', 'Origination Credit Score', 'Filler.4', 'Filler.5', 'Filler.6', 'Origination Loan Purpose', 'Origination Occupancy Status', 'Origination Channel', 'Loan Performance History'

Source: Fannie Mae, Freddie Mac, Internal Analysis

Data Cleaning

A snapshot of data

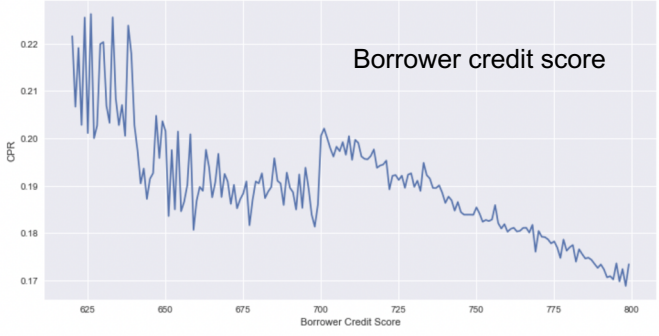
FMAE	Current Net Interest Rate	Loan Term	Mortgage Loan Amount	Current Investor Loan UPB
count	25000	25000	25000	25000
mean	2.5185866	332.49296	279318.68	276183.5099
std	1.225308458	62.1681139	152302.5723	154713.9017
min	0.66	96	13000	233.05
25%	2	360	161000	158000
50%	2	360	250000	249000
75%	2.5	360	371000	370000
max	13	360	1347000	1345000

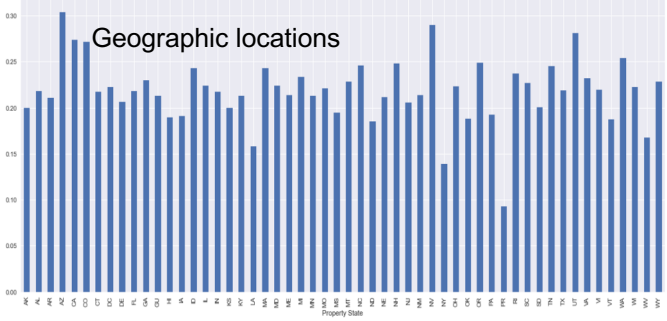
FMAC	Current Net Interest Rate	Loan Term	Mortgage Loan Amount	Current Investor Loan UPB
count	25000	25000	25000	25000
mean	2.17088	321.24576	295788.6	284493.0283
std	0.407684527	70.84960345	136979.4757	134340.0174
min	1.5	96	10000	374.54
25%	2	348	193000	184039.7075
50%	2	360	275000	263876.23
75%	2.5	360	380000	366356.9475
max	7.5	360	1360000	1329836.38

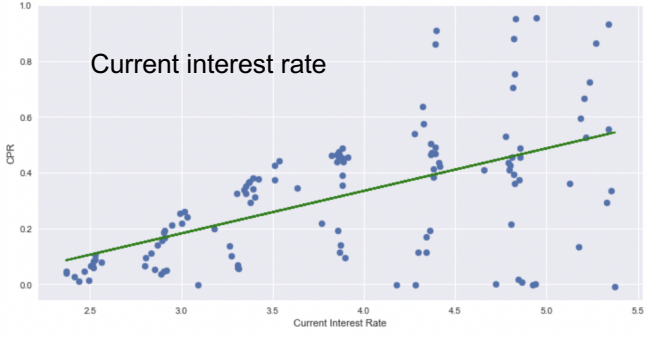
Source: Fannie Mae, Freddie Mac, Internal Analysis

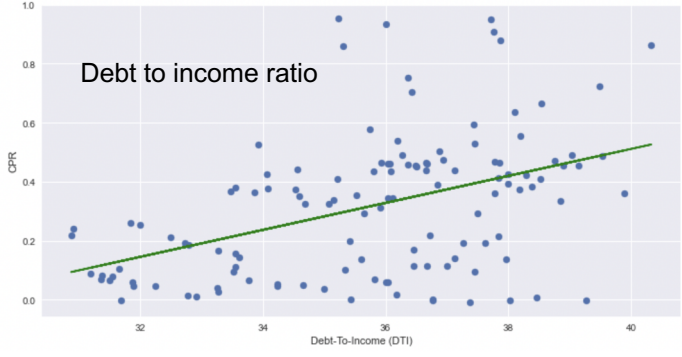
Visualization - variable selection

There are many possible input variables that are related to macroeconomic factors, mortgage characteristics, or clients' specifics.









Variable We Selected

There are many reasons why borrowers decide to make prepayments on their mortgage loan. The factors might be related to borrowers' personal characteristics such as age, income, credit score, loan characteristics such as loan age, interest rate, prepayment penalty, and property type, and macroeconomic data such as month of the year, housing prices.

Given the prepayment prediction requirements, a mix of different parameters is needed

Borrowers' personal characteristics		Loan characteristics		Macroeconomic data	
First time home buyer	✓	Loan purpose	✓	Original Interest rate	✓
Occupancy Status	✓	Chanel	✓	State	✓
Debt to income	✓	Number of borrowers	✓		
Borrower credit score	✓	Loan Term	✓		
		Loan Age	✓		
		Loan to value	✓		

Source: Fannie Mae, Freddie Mac, Internal Analysis

CPR Function

```
def enrich_monthly_principle_payment(df):
    df["Monthly Interest Rate"] = df["Current Net Interest Rate"]/1200
    df["I*( 1 + I )^N"] = df["Monthly Interest Rate"]*(1+df["Monthly Interest Rate"])**df["Loan Term"]
    df["( 1 + I )^N-1"] = (1+df["Monthly Interest Rate"])**(df["Loan Term"])-1
    df["Monthly payment"] = (df["Mortgage Loan Amount"]*df["I*( 1 + I )^N"])/df["( 1 + I )^N-1"]
    df["Monthly interest"] = df["Current Investor Loan UPB"]*df["Monthly Interest Rate"]
    df["Monthly Principal Payment"] = df["Monthly payment"] - df["Monthly interest"]
    return df
```

```
def get_cpr(df1,df2, level):
    df1_combo = df1[[level,"Current Investor Loan UPB","Monthly Principal Payment"]]
    df2_combo = df2[[level,"Current Investor Loan UPB","Monthly Principal Payment"]]
    df1_group = df1_combo.groupby(level).sum()
    df2_group = df2_combo.groupby(level).sum()
    df_total = pd.merge(df1_group, df2_group,on=level)
    df_total["Scheduled Principle Balance"] = df_total["Current Investor Loan UPB_x"] - df_total
    ["Monthly Principal Payment_x"]
    df_total["Prepayment"] = df_total["Scheduled Principle Balance"] - df_total["Current Investor Loan UPB_y"]
    df_total["SMM"] = df_total["Prepayment"]/df_total["Scheduled Principle Balance"]
    df_total["CPR"] = ((1-(1-df_total["SMM"])**12))
    df_total =  df_total.reset_index()
    df_total_result = df_total[[level,"CPR"]]

    return df_total_result
```

CPR_Calculation_202109-202202

CUSIP	CPR_2109-2110	CPR_2110-2111	CPR_2111-2112	CPR_2112-2201	CPR_2201-2202
31418D2H4	0.435949106987052	0.909312505815658	0.527479045059866	0.976255537737526	0.737414775949074
31418D2K7	0.0147990100465649	0.0354690902850321	0.0428201633211005	0.0594742287341895	0.0109965308694537
31418D2L5	0.0467678586246162	0.0410457468427238	0.0565992219893886	0.0632880503782096	0.0727035665079465
31418D2M3	0.0699399898039127	0.0756027028743266	0.106577526676178	0.146304557646543	0.133354642551635
31418D2N1	0.11517529381125	0.146190616891934	0.204349701124197	0.294957866738373	0.347366416352071
31418D2P6	0.116098155868009	0.373455011109816	0.270140271968024	0.419744688590597	0.145873930435118
31418D3D2	0.409134624366465	-0.00159712400706824	0.00303251216761935	0.430153117991165	0.611372634494054

Dataframe Function

Aggregate Independent Variables

```
cat_variable = ['Property State','First Time Home Buyer Indicator','Loan Purpose',
                'Occupancy Status','Channel','Property Valuation Method']

data_need1=pd.DataFrame()
for i in alist:
    subdata = data[data['CUSIP'] == i]
    df = pd.DataFrame({'CUSIP':[i]})
    for x in cat_variable:
        a = subdata.groupby(x)['Loan Identifier'].count()
        a = a.to_frame().transpose().reset_index(drop=True)
        a = a/len(subdata)
        df = df.join(a)
    data_need1=data_need1.append(df,ignore_index=True)
```

```
num_variable = ['Original Interest Rate','Loan Term','Loan Age','Loan-To-Value (LTV)',
                'Debt-To-Income (DTI)','Borrower Credit Score','Number of Borrowers',
                'Mortgage Insurance Percent', "SMM", "CPR"]

data_need2=pd.DataFrame()
for i in alist:
    subdata = data[data['CUSIP'] == i]
    df = pd.DataFrame({'CUSIP':[i]})
    for i in num_variable:
        value = subdata[i].mean()
        b = pd.DataFrame({i:[value]})
        df = df.join(b)
    data_need2=data_need2.append(df,ignore_index=True)
```

Add Mortgage Rate (PMM)

	Date	30 YR FRM	30 Yr Fees & Points
0	2021-09-02	2.87	0.6
1	2021-09-09	2.88	0.7
2	2021-09-16	2.86	0.7
3	2021-09-23	2.88	0.7
4	2021-09-30	3.01	0.7

	Month	30 YR FRM
0	1	3.4450
1	2	3.7625
2	9	2.9000
3	10	3.0675
4	11	3.0675
5	12	3.0980

Dataframe Function

```
def get_data(data, cpr):
    data = data.merge(cpr, on='CUSIP')
    data = data.dropna(axis=1, how='all', thresh=None, subset=None, inplace=False)
    data['First Time Home Buyer Indicator'] = data['First Time Home Buyer Indicator'].replace(['Y','N'],['Yes','No'])
    data['Loan Purpose'] = data['Loan Purpose'].replace(['C','R','P','U'],['Cash-Out','Refinance','Purchase',
                                                                           'Not specified'])
    data['Occupancy Status'] = data['Occupancy Status'].replace(['P','S','I','U'],['Principal','Second','Investor',
                                                                                   'Unknown'])
    data['Channel'] = data['Channel'].replace(['R','C','B'],['Retail','Correspondent','Broker'])

    state = ["NY", "TX", "CA", "IL", "FL"]
    data = data[data["Property State"].isin(state)]

    cat_variable = ['Property State','First Time Home Buyer Indicator','Loan Purpose','Occupancy Status',
                'Channel']
    data_need1=pd.DataFrame()
    alist = []
    for i in data["CUSIP"].unique():
        alist.append(i)
    for i in alist:
        subdata = data[data['CUSIP'] == i]
        df = pd.DataFrame({'CUSIP':[i]})
        for x in cat_variable:
            a = subdata.groupby(x)[ 'Loan Identifier'].count()
            a = a.to_frame().transpose().reset_index(drop=True)
            a = a/len(subdata)
            df = df.join(a)
        data_need1=data_need1.append(df,ignore_index=True)

    num_variable = ['Original Interest Rate','Loan Term','Loan Age','Loan-To-Value (LTV)','Debt-To-Income (DTI)',
                'Borrower Credit Score','Number of Borrowers','Mortgage Insurance Percent', "SMM", "CPR"]
    data_need2=pd.DataFrame()
    for i in alist:
        subdata = data[data['CUSIP'] == i]
        df = pd.DataFrame({'CUSIP':[i]})
        for i in num_variable:
            value = subdata[i].mean()
            b = pd.DataFrame({i:[value]})
            df = df.join(b)
        data_need2=data_need2.append(df,ignore_index=True)

    data_need3 = pd.merge(data_need1, data_need2, on="CUSIP")
    return data_need3
```

Dataframe Function

	CUSIP	CA	FL	IL	NY	TX	No	Yes	Cash-Out	N
0	31418D3X8	0.772696128811237	0.0854744775608085	0.0427657873701039	0.0191846522781775	0.0798789539796734	0.866735183281946	0.133264816718054	0.321400022838872	0.420520726276122
1	31418D3Y6	0.699276694863308	0.0882677454946672	0.0861836459482653	0.036533039107515	0.0897388745862449	0.840137305381881	0.159862694618119	0.477136202035062	0.253769768297168
2	31418D4V1	0.411764705882353	0.352941176470588			0.235294117647059	1		0.705882352941177	0.0588235294117647
3	31418D4A7	0.365853658536585	0.121951219512195	0.109756097560976	0.0487804878048781	0.353658536585366	0.951219512195122	0.0487804878048781	0.695121951219512	0.158536585365854
4	31418D3W0	0.813559322033898	0.0920096852300242	0.0121065375302663	0.0121065375302663	0.0702179176755448	0.949152542372881	0.0508474576271186	0.208232445520581	0.690072639225182
5	31418D3Z3	0.491525423728814	0.116222760290557	0.0968523002421307	0.0653753026634383	0.230024213075061	0.87409200968523	0.12590799031477	0.641646489104116	0.162227602905569
6	31418D3H3	0.684440559440559	0.102272727272727	0.097027972027972	0.020541958041958	0.0957167832167832	0.815268065268065	0.184731934731935	0.411276223776224	0.270541958041958

Purchase	Investor	Principal	Second	Broker	Correspondent	Retail	Original Interest Rate	Loan Term	Loan Age
0.258079250885006	0.00519584332533973	0.976019184652278	0.0187849720223821	0.465627498001599	0.187792623044422	0.34657987895398	2.8722578508622	360	1.97299303414411
0.26909402966777	0.0134853500061297	0.962731396346696	0.0237832536471742	0.195292386906951	0.18585264190266	0.618854971190389	3.31084712516856	360	2.09341669731519
0.235294117647059	0.352941176470588	0.588235294117647	0.0588235294117647	0.0588235294117647	0.294117647058824	0.647058823529412	4.875	360	2.35294117647059
0.146341463414634	0.134146341463415	0.841463414634146	0.024390243902439	0.0853658536585366	0.219512195121951	0.695121951219512	4.34268292682927	360	2.1219512195122
0.101694915254237		0.987893462469734	0.0121065375302663	0.498789346246973	0.278450363196126	0.222760290556901	2.37064648910412	360	2.10653753026634
0.196125907990315	0.0411622276029056	0.946731234866828	0.0121065375302663	0.0460048426150121	0.302663438256659	0.651331719128329	3.89939467312349	360	2.19128329297821

Loan-To-Value (LTV)	Debt-To-Income (DTI)	Borrower Credit Score	Number of Borrowers	Mortgage Insurance Percent	SMM	CPR
63.3058124928629	36.4177229644856	759.118476647254	1.47527692132009	3.63080963800388	0.00192729045236141	0.0228838999602214
70.4768910138531	37.470516121123	737.322422459237	1.50508765477504	5.0289322054677	0.00258110729577028	0.0305373491274863
70.8823529411765	34.9411764705882	693.882352941176	1.58823529411765	2.17647058823529	0.0394514614494468	0.383075828025749
74.2439024390244	38.6585365853659	689.378048780488	1.4390243902439	2.8780487804878	0.0017090076572054	0.020316419134402
56.6174334140436	34.3462469733656	768.302663438257	1.47215496368039	1.58353510895884	8.17021082347368E-05	0.000979984853302598
71.682808716707	38.4261501210654	701.157384987893	1.49636803874092	3.84503631961259	0.0121945813108407	0.136908478679574

Models

Our Rationale

Machine Learning

1. We first used Month t's SMM to regress on Month (t-1)'s independent variables
    - CA,FL,IL,NY,TX,No,Yes,Cash-Out,Purchase,Investor,Principal,Second,Broker,Correspondent,Retail,Original Interest Rate,Loan Term,Loan Age,Loan-To-Value (LTV),Debt-To-Income (DTI),Borrower Credit Score,Number of Borrowers,Mortgage Insurance Percentage
2. We then used January's data to predict the SMM in February, and calculated the corresponding accuracy of our model accordingly.
3. We then used February's predicted SMM to calculate February's predicted CPR and compared it to the actual CPR in February.
4. Finally, we used February's data to predict the SMM and CPR of March.

Source: Fannie Mae, Freddie Mac, Internal Analysis

Linear Regression

Data loading and processing

1

```
.set_option('display.max_columns', None)
df = df.dropna(axis=1, how='all', thresh=None, subset=None, inplace=False)
df
```

id	Broker	Correspondent	Retail	Original Interest Rate	Loan Term	Loan Age	Loan-To-Value (LTV)	Debt-To-Income (DTI)	Borrower Credit Score	Number of Borrowers	Mortgage Insurance Percent	SMM	CPR	Month	PMM
70	0.465025	0.188188	0.346788	2.872365	360	0.973662	63.298615	36.416681	759.119719	1.475058	3.630751	0.003131	0.036937	October	3.0675
52	0.195283	0.185995	0.618722	3.310758	360	1.093975	70.481364	37.470121	737.421850	1.504949	5.030062	0.005066	0.059132	October	3.0675
24	0.058824	0.294118	0.647059	4.875000	360	1.352941	70.882353	34.941176	693.882353	1.588235	2.176471	0.001423	0.016946	October	3.0675
14	0.083333	0.226190	0.690476	4.341964	360	1.130952	74.380952	38.785714	689.773809	1.440476	2.809524	0.017708	0.192979	October	3.0675
2	498789	0.278450	0.222760	2.370646	360	1.106538	56.617433	34.346247	768.302663	1.472155	1.612591	0.003351	0.039477	October	3.0675
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
									319	1.5633	5.135371	0.049407	0.455582	February	3.7625
									341	1.5384	.474359	0.031646	0.320152	February	3.7625

2

```
df['Month'].loc[df['Month']=='December']=12
df['Month'].loc[df['Month']=='November']=11
df['Month'].loc[df['Month']=='October']=10
df['Month'].loc[df['Month']=='January']=13
df['Month'].loc[df['Month']=='February']=14
df.sort_values(by=['CUSIP','Month'],inplace=True)
df
```

```
df['CPR'] = df['CPR'].shift(-1)
df['SMM'] = df['SMM'].shift(-1)
df
```

3

```
df.set_index('CUSIP', inplace=True)
df_tt = df[df['Month']<13]
df_tt
```

```
df_vali=df[df['Month']==13]
df_vali
```

Source: Fannie Mae, Freddie Mac, Internal Analysis

Linear Regression

Model Training

```
x_train = df_tt.drop(columns = ['Month','SMM','CPR'],axis=1)
y_train = df_tt['SMM']
x_test = df_vali.drop(columns = ['Month','SMM','CPR'],axis=1)
y_test = df_vali['SMM']


print(f'x_train.shape = {x_train.shape}')
print(f'y_train.shape = {y_train.shape}')
print('\n')
print(f'x_test.shape  = {x_test.shape}')
print(f'y_test.shape  = {y_test.shape}')
```

```
x_train.shape = (375, 25)
y_train.shape = (375,)


x_test.shape  = (125, 25)
y_test.shape  = (125,)
```

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)

coeff_df = pd.DataFrame(regressor.coef_, x_train.columns, columns=['Coefficient'])
coeff_df
```

	Coefficient
CA	-4.531259e+06
FL	-4.531259e+06
IL	-4.531259e+06
NY	-4.531259e+06
TX	-4.531259e+06
No	-9.467157e-01
Yes	-8.846785e-01
Cash-Out	-7.716777e+05
N	-7.716777e+05
Purchase	-7.716778e+05
Investor	-1.917803e+06
Principal	-1.917803e+06
Second	-1.917803e+06
Broker	-3.847233e+06
Correspondent	-3.847233e+06
Retail	-3.847233e+06
Original Interest Rate	3.329437e-02
Loan Term	-1.763293e-03
Loan Age	8.341390e-05
Loan-To-Value (LTV)	8.574156e-04
Debt-To-Income (DTI)	1.738015e-06
Borrower Credit Score	1.291154e-04
Number of Borrowers	6.956458e-02
Mortgage Insurance Percent	-3.975854e-03
PMM	1.175107e-01

Source: Fannie Mae, Freddie Mac, Internal Analysis

Linear Regression

Model Testing and data prediction

```
y_pred = regressor.predict(x_test)
df_model = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_model
```

CUSIP	Actual	Predicted
31418D2H4	0.105447	0.108603
31418D2K7	0.000921	0.044251
31418D2L5	0.006270	0.034089
31418D2M3	0.011856	0.050485
31418D2N1	0.034937	0.094737
...	...	...
31418DZU9	0.004414	0.062237
31418DZV7	0.004996	0.040965
31418DZW5	0.012451	0.057822
31418DZX3	0.026853	0.083902
31418DZY1	0.008056	0.078751

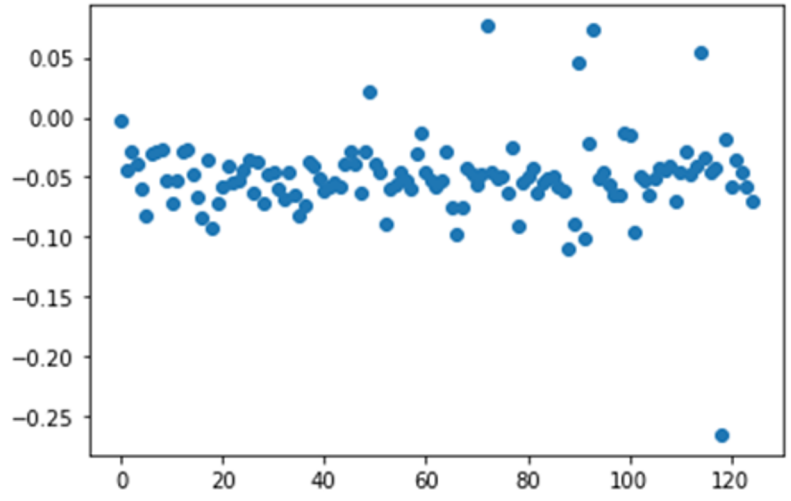
125 rows × 2 columns

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('R-squared:', compute_r2(y_test,y_pred))
```

```
Mean Absolute Error: 0.053538264372929883
Mean Squared Error: 0.003593487803754699
Root Mean Squared Error: 0.05994570713366136
R-squared: (-2.619500230738914, 0.44918597546933725, 0.12410165681289011)
```

```
from matplotlib import pyplot as plt

plt.rcParams['agg.path.chunksize'] = 20000
plt.scatter(range(len(y_test)),(y_test-y_pred))
plt.show()
```



Source: Fannie Mae, Freddie Mac, Internal Analysis

Linear Regression

CPR Deduction

```
# calculate CPR  CPR = (1 - (1 - SMM)^12)*100

df_cpr_vali = pd.DataFrame({'Actual_CPR': y_test, 'Predicted_CPR':y_pred })
df_cpr_vali['Actual_CPR']=(1-(1-df_cpr_vali['Actual_CPR'])**12)*100
df_cpr_vali['Predicted_CPR']=(1-(1-df_cpr_vali['Predicted_CPR'])**12)*100
df_cpr_vali['Deviation']=abs(df_cpr_vali['Predicted_CPR']-df_cpr_vali['Actual_CPR'])/df_cpr_vali['Actual_CPR']
df_cpr_vali
```

CUSIP	Actual_CPR	Predicted_CPR	Deviation
31418D2H4	73.741478	74.831646	0.014784
31418D2K7	1.099653	41.906788	37.109094
31418D2L5	7.270357	34.045504	3.682783
31418D2M3	13.335464	46.293930	2.471490
31418D2N1	34.736641	69.709946	1.006813
...	...	...	...
31418DZU9	5.170432	53.749354	9.395526
31418DZV7	5.833280	39.464259	5.765363
31418DZW5	13.958974	51.067777	2.658419
31418DZX3	27.865762	65.061750	1.334828
31418DZY1	9.250383	62.629884	5.770518

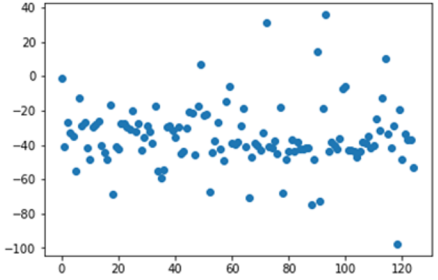
125 rows × 3 columns

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predicted_CPR']))
print('Mean Squared Error:', metrics.mean_squared_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predicted_CPR']))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predicted_CPR'])))
```

```
Mean Absolute Error: 36.49026179575191
Mean Squared Error: 1553.5256654851185
Root Mean Squared Error: 39.4147899332867
```

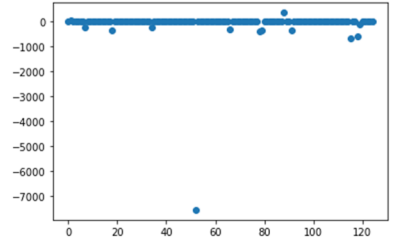
```
from matplotlib import pyplot as plt

plt.rcParams['agg.path.chunksize'] = 20000
plt.scatter(range(len(df_cpr_vali['Actual_CPR'])),(df_cpr_vali['Actual_CPR']-df_cpr_vali['Predicted_CPR']))
plt.show()
```




```
from matplotlib import pyplot as plt

plt.rcParams['agg.path.chunksize'] = 20000
plt.scatter(range(len(df_cpr_vali['Actual_CPR'])),df_cpr_vali['Deviation'])
plt.show()
```



Source: Fannie Mae, Freddie Mac, Internal Analysis

Linear Regression

SMM and CPR for March

```
# finally predict the SMM and CPR for March
df_Feb= df[df['Month']==14]
x_Feb=df_Feb.drop(columns = ['Month','SMM','CPR'],axis=1)
y_pred_March = regressor.predict(x_Feb)
df_pred_March = pd.DataFrame({'CUSIP':df_Feb.index, 'Predicted SMM': y_pred_March, 'Predicted CPR':(1-(1-y_pred_March)**12)*100 })
df_pred_March
```

	CUSIP	Predicted SMM	Predicted CPR
0	31418D2H4	0.113230	76.355363
1	31418D2K7	0.079521	63.002926
2	31418D2L5	0.069658	57.955094
3	31418D2M3	0.092437	68.773719
4	31418D2N1	0.140003	83.633107
...	...	...	...
120	31418DZU9	0.094240	69.510117
121	31418DZV7	0.086798	66.364327
122	31418DZW5	0.096217	70.299137
123	31418DZX3	0.119403	78.256556
124	31418DZY1	0.105641	73.809606

125 rows × 3 columns

Source: Fannie Mae, Freddie Mac, Internal Analysis

Decision Trees

Rationale for using Decision Trees Model

- heterogeneous data
- non parametric
- easiness of interpretation

1. Read data and process

2. Model training

```
x_train = df_tt.drop(columns = ['Month','SMM','CPR'],axis=1)
y_train = df_tt['SMM']
x_test = df_vali.drop(columns = ['Month','SMM','CPR'],axis=1)
y_test = df_vali['SMM']


print(f'x_train.shape = {x_train.shape}')
print(f'y_train.shape = {y_train.shape}')
print('\n')
print(f'x_test.shape  = {x_test.shape}')
print(f'y_test.shape  = {y_test.shape}')
```

```
x_train.shape = (375, 25)
y_train.shape = (375,)


x_test.shape  = (125, 25)
y_test.shape  = (125,)
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree
model = tree.DecisionTreeRegressor()
model.fit(x_train,y_train)
```

```
DecisionTreeRegressor()
```

3. Model testing

```
y_pred = model.predict(x_test)
df_model = pd.DataFrame({'Actual SMM': y_test, 'Predicted SMM': y_pred})
df_model
```

CUSIP	Actual SMM	Predicted SMM
31418D2H4	0.105447	-0.000127
31418D2K7	0.000921	0.005097
31418D2L5	0.006270	0.005433
31418D2M3	0.011856	0.006530
31418D2N1	0.034937	0.028705
...	...	...
31418DZU9	0.004414	0.003762
31418DZV7	0.004996	0.006797
31418DZW5	0.012451	0.015999
31418DZX3	0.026853	0.036002
31418DZY1	0.008056	0.089881

125 rows × 2 columns

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.023785562343572646
Mean Squared Error: 0.0028386848540292373
Root Mean Squared Error: 0.053279309811870096
```

Source: Fannie Mae, Freddie Mac, Internal Analysis

Decision Trees

4. SMM Prediction and CPR calculation

```
df_cpr_vali = pd.DataFrame({'Actual_CPR': y_test, 'Predicted_CPR':y_pred })
df_cpr_vali['Actual_CPR']=(1-(1-df_cpr_vali['Actual_CPR'])**12)*100
df_cpr_vali['Predicted_CPR']=(1-(1-df_cpr_vali['Predicted_CPR'])**12)*100
df_cpr_vali['Error Percentage']= abs((df_cpr_vali['Predicted_CPR']-df_cpr_vali['Actual_CPR'])/df_cpr_vali['Actual_C
df_cpr_vali
```

CUSIP	Actual_CPR	Predicted_CPR	Error Percentage
31418D2H4	73.741478	-0.152107	1.002063
31418D2K7	1.099653	5.947423	4.408454
31418D2L5	7.270357	6.328805	0.129506
31418D2M3	13.335464	7.560270	0.433070
31418D2N1	34.736642	29.495787	0.150874
...	...	...	...
31418DZU9	5.170432	4.422358	0.144683
31418DZV7	5.833280	7.858628	0.347206
31418DZW5	13.958974	17.596435	0.260582
31418DZX3	27.865763	35.596219	0.277418
31418DZY1	9.250382	67.701852	6.318817

```
df_Feb= data[data['Month']==14]
x_Feb=df_Feb.drop(columns = ['Month','SMM','CPR'],axis=1)
y_pred_March = model.predict(x_Feb)
df_pred_March = pd.DataFrame({ 'Predicted SMM': y_pred_March, 'Predicted CPR':(1-(1-y_pred_March)**12)*100 })
df_pred_March
```

	Predicted SMM	Predicted CPR
0	0.123965	79.570568
1	0.005097	5.947423
2	0.005433	6.328805
3	0.013095	14.630456
4	0.018869	20.434970
...	...	...
120	0.003762	4.422358
121	0.006797	7.858628
122	0.015999	17.596435
123	0.039563	38.393681
124	0.060561	52.747905

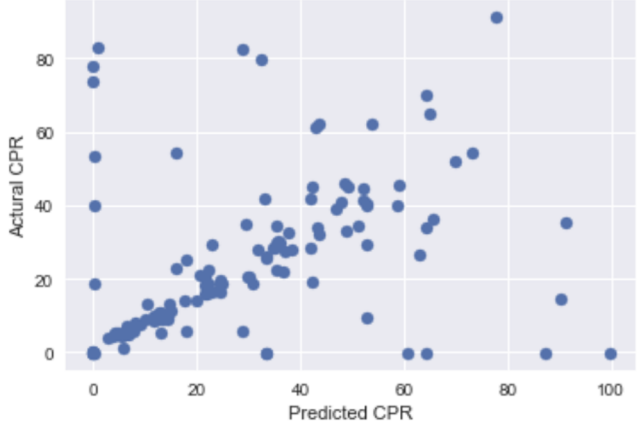
Decision Trees

5. Model Evaluation

	Actual_CPR	Predicted_CPR	Error Percentage
count	125.000000	125.000000	125.000000
mean	24.621853	29.633955	17.623759
std	21.103340	23.181953	87.420949
min	-0.241265	-0.240824	0.000906
25%	8.101506	10.535739	0.194585
50%	19.648068	24.957440	0.351626
75%	34.736642	43.348443	0.658206
max	91.212940	99.785483	651.902499

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predicted_CPR']))
print('Mean Squared Error:', metrics.mean_squared_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predicted_CPR']))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(df_cpr_vali['Actual_CPR'], df_cpr_vali['Predic
```

```
Mean Absolute Error: 14.387758597865702
Mean Squared Error: 630.6677905076036
Root Mean Squared Error: 25.11309997805137
```



Random Forest

Model Tuning - GridSearch

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
parameters = {
     'max_depth':(2,4,6,8),
     'min_samples_leaf': (2,4,6,8),
     'min_samples_split': (2, 4, 8),
     'n_estimators':(5, 10, 15) #the number of trees
}

model = GridSearchCV(RandomForestRegressor(),parameters,cv=3)
model.fit(x_train, np.ravel(y_train))
model.best_params_
```

max_depth	min_samples_leaf	min_samples_split	n_estimators
4	4	2	10

Training Model

```
model = RandomForestRegressor(max_depth=4,
                              min_samples_leaf=4,
                              min_samples_split=2,
                              n_estimators=10)
model.fit(x_train,np.ravel(y_train))
```

Training Results

R Squared	0.077435
Mean Absolute Error	0.018612
Mean Squared Error	0.000916
Root Mean Squared Error	0.030264

Source: Fannie Mae, Freddie Mac, Internal Analysis

Random Forest

Model Testing

*Error Percentage = abs(actual CPR - predict CPR) / actual CPR

CUSIP	Actual_CPR	Predicted_CPR	Error Percentage
31418D2H4	0.737415	0.494999	0.328738
31418D2K7	0.010997	0.055987	4.091347
31418D2L5	0.072704	0.071299	0.019322
31418D2M3	0.133355	0.111292	0.165445
31418D2N1	0.347366	0.340174	0.020705

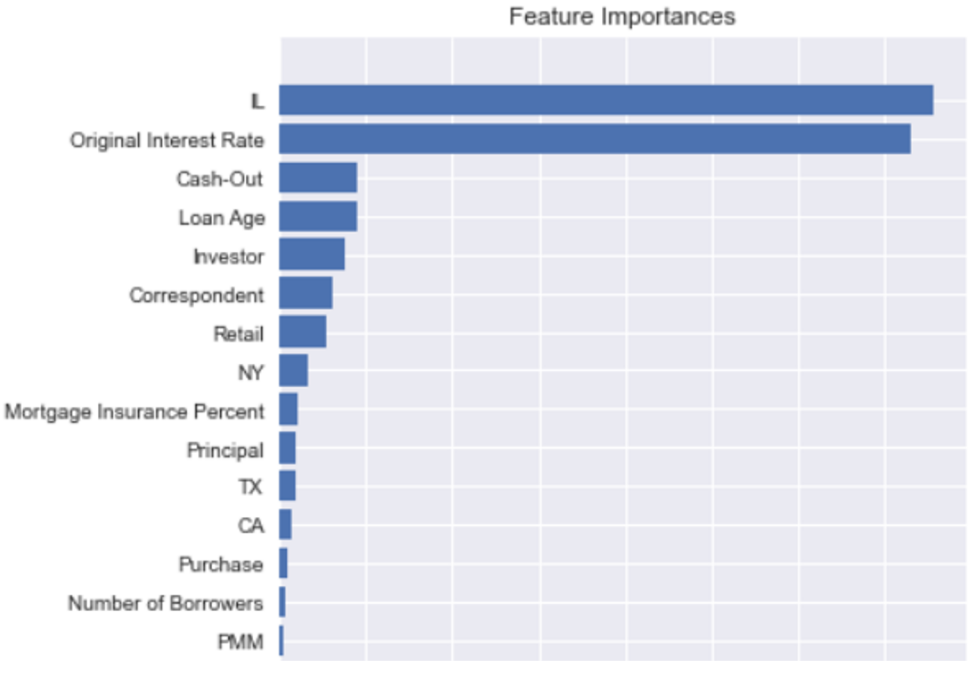
Testing Results

Mean Absolute Error	0.135698
Mean Squared Error	0.040438
Root Mean Squared Error	0.201093
Mean Error Percentage	0.270127

Source: Fannie Mae, Freddie Mac, Internal Analysis

Random Forest

Model Feature Importance



Predicted SMM & CPR

	CUSIP	Predicted SMM	Predicted CPR
0	31418D2H4	0.065986	0.559199
1	31418D2K7	0.005208	0.060731
2	31418D2L5	0.006791	0.078513
3	31418D2M3	0.012881	0.144081
4	31418D2N1	0.031499	0.318919
...	...	...	...
120	31418DZU9	0.005792	0.067334
121	31418DZV7	0.006575	0.076110
122	31418DZW5	0.014963	0.165497
123	31418DZX3	0.036830	0.362566
124	31418DZY1	0.067519	0.567806

Source: Fannie Mae, Freddie Mac, Internal Analysis

Neural Networks

We tried different Parameters

Neural networks model can produce highly accurate fits to the historical prepayment trends and has the capacity to learn relationships from past data. It is also able to model large groups of risk measures that are potentially nonlinear and highly interactive.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout
```

```
nn_model2 = Sequential()

nn_model2.add(Dense(units=78,activation='relu'))
nn_model2.add(Dropout(0.5))
nn_model2.add(Dense(units=39,activation='relu'))
nn_model2.add(Dropout(0.5))
nn_model2.add(Dense(units=1,activation='sigmoid'))
nn_model2.compile(loss='binary_crossentropy',optimizer='adam')
```

```
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=25)
```

…to come up with SMM predictions

```
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=25)
```

```
nn_model2.fit(x=x_train,y=y_train,epochs=200,validation_data=(x_test,y_test)
          ,verbose=1,callbacks=[early_stop],batch_size=256)
```

```
Epoch 1/200
2/2 [==============================] - 1s 116ms/step - loss: 0.7822 - val_loss: 0.8021
Epoch 2/200
2/2 [==============================] - 0s 23ms/step - loss: 0.7085 - val_loss: 0.5830
Epoch 3/200
2/2 [==============================] - 0s 58ms/step - loss: 0.6423 - val_loss: 0.4085
Epoch 4/200
2/2 [==============================] - 0s 28ms/step - loss: 0.5885 - val_loss: 0.2852
Epoch 5/200
2/2 [==============================] - 0s 31ms/step - loss: 0.5312 - val_loss: 0.2052
Epoch 6/200
2/2 [==============================] - 0s 37ms/step - loss: 0.5038 - val_loss: 0.1602
Epoch 7/200
2/2 [==============================] - 0s 35ms/step - loss: 0.4377 - val_loss: 0.1374
Epoch 8/200
2/2 [==============================] - 0s 38ms/step - loss: 0.4075 - val_loss: 0.1287
Epoch 9/200
2/2 [==============================] - 0s 31ms/step - loss: 0.3728 - val_loss: 0.1285
```

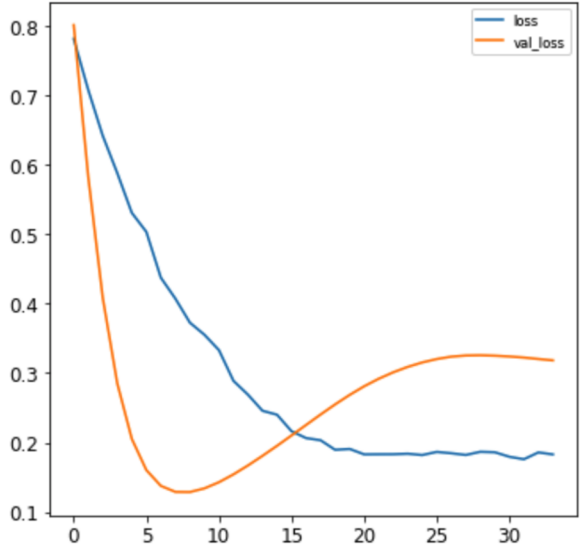
Source: Fannie Mae, Freddie Mac, Internal Analysis

Neural Networks

Graph the loss

```
nn_model_loss=pd.DataFrame(nn_model2.history.history)
nn_model_loss.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa73e3d11c0>
```



…and outcome for SMM

```
nn_pred2 = nn_model2.predict(x_test)
nn_pred2 = nn_pred2.flatten()
```

```
nn2_model = pd.DataFrame({'Actual SMM': y_test, 'Predicted SMM': nn_pred2})
nn2_model
```

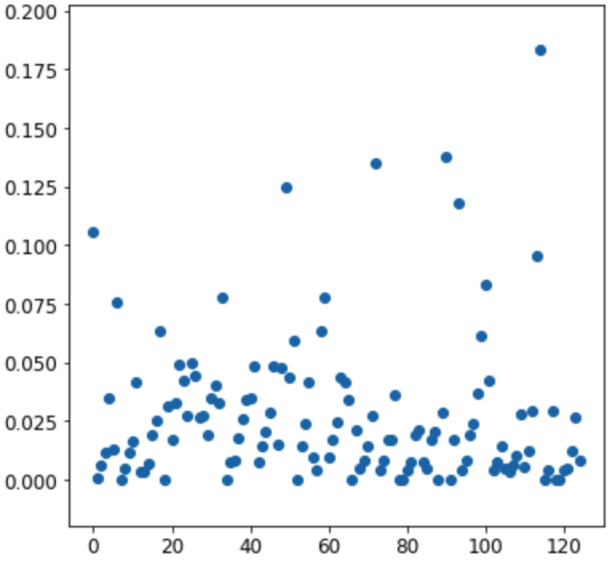
CUSIP	Actual SMM	Predicted SMM
31418D2H4	0.105447	0.000018
31418D2K7	0.000921	0.000009
31418D2L5	0.006270	0.000008
31418D2M3	0.011856	0.000008
31418D2N1	0.034937	0.000012
...	...	...
31418DZU9	0.004414	0.000010
31418DZV7	0.004996	0.000008
31418DZW5	0.012451	0.000008

Source: Fannie Mae, Freddie Mac, Internal Analysis

Neural Networks

Model Testing

Mean Absolute Error: 0.027750483613300656
Mean Squared Error: 0.0017614652406482197
Root Mean Squared Error: 0.04196981344547793



Comparing Actuals and Predicted

```
df_cpr_vali2 = pd.DataFrame({'Actual_CPR': y_test, 'Predicted_CPR':nn_pred2 })
df_cpr_vali2['Actual_CPR']=1-(1-df_cpr_vali2['Actual_CPR'])**12
df_cpr_vali2['Predicted_CPR']=1-(1-df_cpr_vali2['Predicted_CPR'])**12

df_cpr_vali2
```

CUSIP	Actual_CPR	Predicted_CPR
31418D2H4	0.737415	0.000222
31418D2K7	0.010997	0.000109
31418D2L5	0.072704	0.000099
31418D2M3	0.133355	0.000097
31418D2N1	0.347366	0.000147
...	...	...
31418DZU9	0.051704	0.000117
31418DZV7	0.058333	0.000099
31418DZW5	0.139590	0.000099
31418DZX3	0.278658	0.000132
31418DZY1	0.092504	0.000149

Source: Fannie Mae, Freddie Mac, Internal Analysis

Gradient Descent

We tried different parameters to tune the model

```
# xgb models only accept 'DMatrix' input; convert the data here
dtrain = xgb.DMatrix(x_train, label=y_train)
dtest = xgb.DMatrix(x_test, label=y_test)

# parameter grid for xgb model
param_dict = {
    # parameters and tuning
    'max_depth':6,
    'min_child_weight': 1,
    'eta':.1,
    'subsample': 1,
    'colsample_bytree': 1,
    # Other parameters
    'objective':'reg:linear',
    'eval_metric':'rmse'
}

# train XGB model on split training data using split test data
num_boost_round=999
xgb_model = xgb.train(
    param_dict,
    dtrain,
    num_boost_round=num_boost_round,
    evals=[(dtest, "Test")],
    early_stopping_rounds=10
)

print("Model's best RMSE on test set: ", xgb_model.best_score)
print("Model's best iteration: ", xgb_model.best_iteration+1)
```

…and yielded the following results

Gradient descent is an optimization algorithm that finds the optimal weights (a,b) that reduces prediction error.

```
df_model = pd.DataFrame({'Actual SMM': y_test, 'Predicted SMM': GBDT_test_pred})
df_model
```

CUSIP	Actual SMM	Predicted SMM
31418D2H4	0.105447	0.141843
31418D2K7	0.000921	0.005229
31418D2L5	0.006270	0.012185
31418D2M3	0.011856	0.016255
31418D2N1	0.034937	0.022836
...	...	...
31418DZU9	0.004414	0.004720
31418DZV7	0.004996	0.008834
31418DZW5	0.012451	0.016574
31418DZX3	0.026853	0.038128
31418DZY1	0.008056	0.056113

125 rows × 2 columns

Source: Fannie Mae, Freddie Mac, Internal Analysis

Gradient Descent

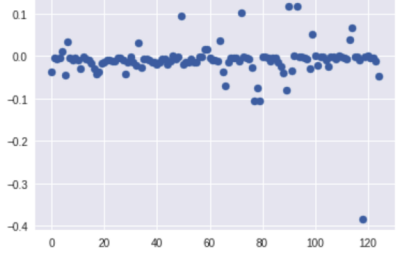
Model Testings

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, GBDT_test_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, GBDT_test_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, GBDT_test_pred)))
```

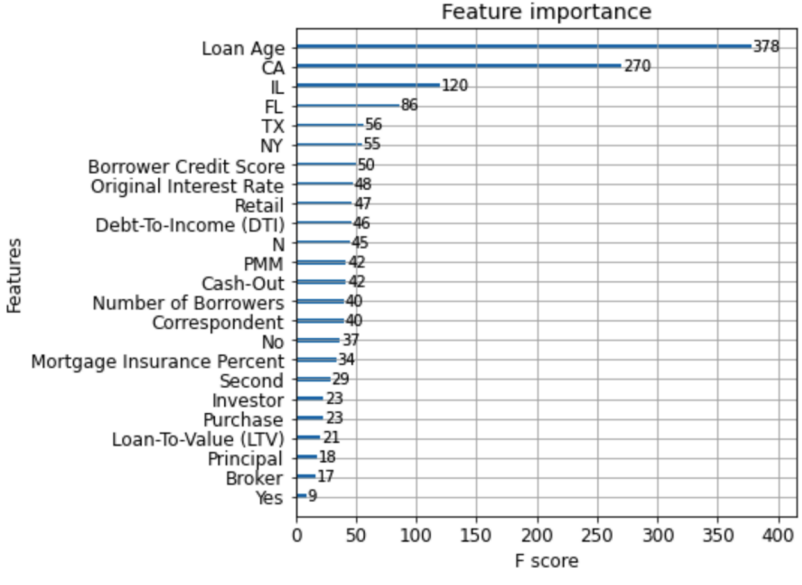
```
Mean Absolute Error: 0.0223211566211291
Mean Squared Error: 0.002199542118955521
Root Mean Squared Error: 0.04689927631590408
```

```
from matplotlib import pyplot as plt

plt.rcParams['agg.path.chunksize'] = 20000
plt.scatter(range(len(y_test)),(y_test-GBDT_test_pred))
plt.show()
```



Model Feature Importance



Source: Fannie Mae, Freddie Mac, Internal Analysis

Gradient Descent

Our Eventual Predictions

```
df_cpr_vali = pd.DataFrame({'Actual_CPR': y_test, 'Predicted_CPR':GBDT_test_pred })
df_cpr_vali['Actual_CPR']=1-(1-df_cpr_vali['Actual_CPR'])**12
df_cpr_vali['Predicted_CPR']=1-(1-df_cpr_vali['Predicted_CPR'])**12

df_cpr_vali
```

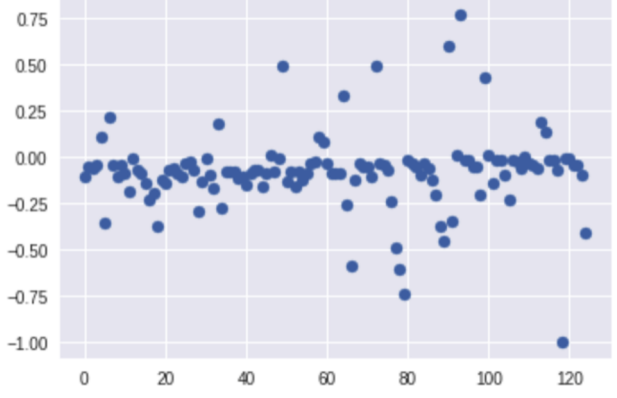
CUSIP	Actual_CPR	Predicted_CPR
31418D2H4	0.737415	0.840484
31418D2K7	0.010997	0.060978
31418D2L5	0.072704	0.136811
31418D2M3	0.133355	0.178531
31418D2N1	0.347366	0.242107
...	...	...
31418DZU9	0.051704	0.055188
31418DZV7	0.058333	0.101002
31418DZW5	0.139590	0.181722
31418DZX3	0.278658	0.372797
31418DZY1	0.092504	0.499920

125 rows × 2 columns

…and Graphing the Difference between Actual and Predicted

```
from matplotlib import pyplot as plt

plt.rcParams['agg.path.chunksize'] = 20000
plt.scatter(range(len(df_cpr_vali['Actual_CPR'])),(df_cpr_vali['Actual_CPR']-df_cpr_vali['Predicted_CPR']))
plt.show()
```



Source: Fannie Mae, Freddie Mac, Internal Analysis

Future Improvements

In order to yield future outcomes in predicting CPR, we suggest the following things:

1	**Incorporate more Macroeconomic Factors**
2	**Generate Individual Behavior Probability base on their Credit Score, etc. to predict likelihood of prepay**
3	**Take into account more month to see the bigger picture**
4	**Tune models with more possible parameters**
5	**Incorporate neural networks with gradient descent**

Our thoughts on the Future Trends in the Mortgage-Backed Securities Universe

Macroeconomics Influencing Factors

Monetary Policy Uncertainties:

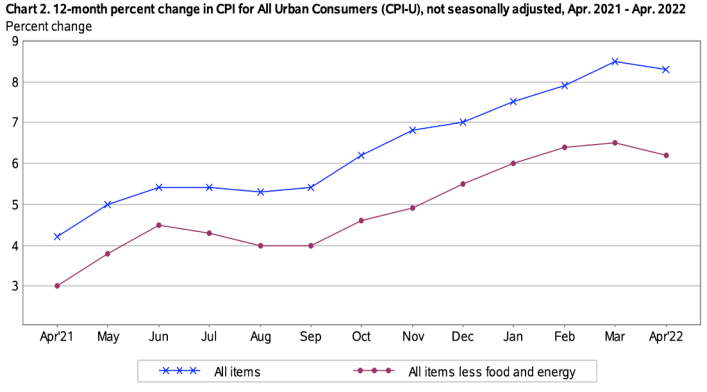
- Quantitative Tightening & Fed reducing the size of the balance sheet
- Economic Uncertainty
- Raise the target range for the federal funds rate

Collateralized | Liquidity | Risk & Return

Market Snapshot

Interest Rate on 15 yr Mortgage 4.88%	
Inflation	8.3%
Unemployment	3.6%

30 Year Fixed Mortgage Rates FY 17-22



Source: BLS, Fannie Mae, Freddie Mac, Internal Analysis

Thank you
Any Questions?

NOMURA

Disclaimer

The content found in this document is for informational and educational purposes only and is not to be considered personal, legal or investment advice or a recommendation to buy or sell securities or any other product.

The content is based on opinions, public filings, current events, press releases and interviews but is not infallible. It may contain errors and writers offer no inferred or explicit warranty as to the accuracy of the information presented.

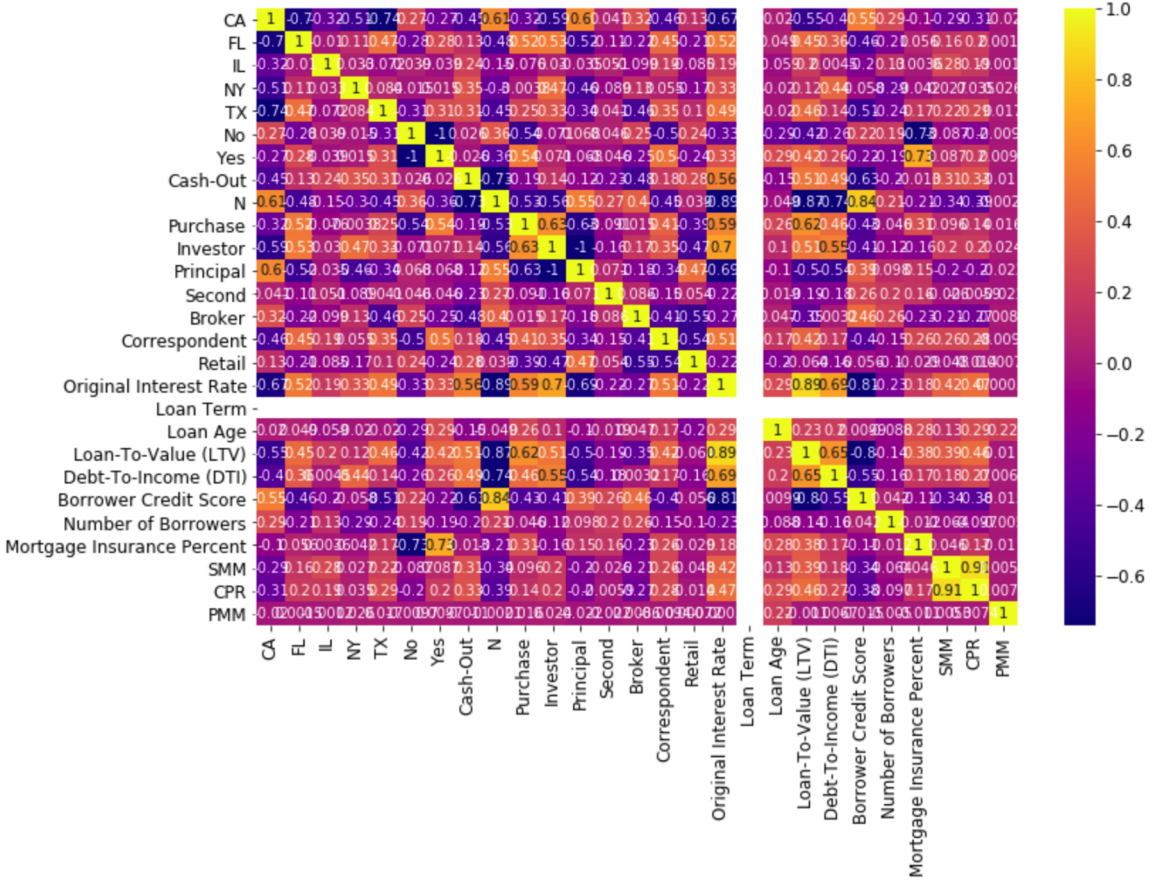
All writers' opinions are their own and do not constitute financial advice in any way whatsoever. Nothing included in this document constitutes an investment recommendation.

By reading this document, you agree to hold the writers, affiliated institutions and partners harmless and to completely release them from any and all liabilities due to any and all losses, damages, or injuries (financial or otherwise) that may be incurred.

Appendix

NOMURA

Correlation Heat Map



Data Summary

	Data Type	Missing Values	Unique Values	count	unique	top	freq	mean	std	min	25%	50%	75%	max
CA	float64	0	483	625.0				0.619134	0.203482	0.0	0.5	0.648649	0.786096	1.0
FL	float64	0	479	625.0				0.104851	0.080901	0.0	0.048503	0.085829	0.138527	0.5
IL	float64	0	469	625.0				0.083245	0.071707	0.0	0.05618	0.074074	0.090909	1.0
NY	float64	0	457	625.0				0.050934	0.084063	0.0	0.014221	0.029412	0.061209	1.0
TX	float64	0	470	625.0				0.141836	0.109743	0.0	0.054832	0.117647	0.208791	0.666667
No	float64	0	466	625.0				0.878523	0.08137	0.607143	0.842105	0.891791	0.92429	1.0
Yes	float64	0	466	625.0				0.121477	0.08137	0.0	0.07571	0.108209	0.157895	0.392857
Cash-Out	float64	0	482	625.0				0.414801	0.194661	0.0	0.255501	0.428571	0.552279	1.0
N	float64	0	471	625.0				0.315176	0.226064	0.0	0.13881	0.269231	0.501567	0.769231
Purchase	float64	0	484	625.0				0.270022	0.158554	0.0	0.16685	0.235294	0.333333	1.0
Investor	float64	0	479	625.0				0.135202	0.184869	0.0	0.008956	0.059379	0.208333	1.0
Principal	float64	0	480	625.0				0.847594	0.182822	0.0	0.790323	0.923497	0.96789	1.0
Second	float64	0	401	625.0				0.017204	0.017367	0.0	0.0	0.01862	0.024175	0.142857
Broker	float64	0	470	625.0				0.231796	0.160815	0.0	0.114943	0.236025	0.318258	1.0
Correspondent	float64	0	488	625.0				0.287963	0.160263	0.0	0.168219	0.252508	0.360902	1.0
Retail	float64	0	493	625.0				0.480241	0.174537	0.0	0.402174	0.480466	0.589091	1.0
Original Interest Rate	float64	0	537	625.0				3.885135	0.879809	2.370646	3.179144	3.878878	4.78125	5.5
Loan Term	float64	0	1	625.0				360.0	0.0	360.0	360.0	360.0	360.0	360.0
Loan Age	float64	0	606	625.0				12.626914	5.850985	0.973662	7.888889	12.731252	17.310391	24.489899

Data Summary

Loan-To-Value (LTV)	float64	0	529	625.0				70.468	6.136218	55.536667	65.883674	71.682809	74.517241	84.909091
Debt-To-Income (DTI)	float64	0	531	625.0				36.896068	2.636695	32.0	34.941176	37.014591	38.578947	50.0
Borrower Credit Score	float64	0	537	625.0				731.765112	31.610632	638.0	704.939394	731.820252	761.428442	795.507143
Number of Borrowers	float64	0	495	625.0				1.482601	0.131164	1.0	1.458824	1.505259	1.525896	2.0
Mortgage Insurance Percent	float64	0	515	625.0				3.776542	2.152898	0.0	2.69697	3.630751	4.937956	14.272727
SMM	float64	1	624	624.0				0.034711	0.041109	-0.001075	0.007714	0.025097	0.046385	0.400731
CPR	float64	1	624	624.0				0.289127	0.230271	-0.012972	0.088735	0.262881	0.434445	0.997855
Month	object	0	5	625.0	5.0	10.0	125.0							
PMM	float64	0	4	625.0				3.2881	0.277039	3.0675	3.0675	3.098	3.445	3.7625