

# LendingClub Analysis

by Charla Gaddy

# About LendingClub

Established in 2007, LendingClub is an online peer-to-peer lending company that brings borrowers and investors together transforming the way borrowers access credit. They provided personal loans, auto refinance, small business loans and patient care loans.

If you want to become an investor you have the choice of a taxable or retirement account. To do this you open an account, browse loans and purchase notes (a fraction of loan with increments as low as \$25).

# The Purpose

The purpose of this project is to create a product which can be used by LendingClub to detect customers who are most likely to not pay back the loan.

# The Plan

- First, we will use Random Forest and XGBoost to predict the bad loans.
- Second, use Kmeans on the top ten features to see if there are any unknown clusters.
- Third, using time series, predict 8 weeks of bad loans and forecast 12 weeks.
- Finally, using LSTM, predict 8 weeks of bad loans.

# The Data

All data was downloaded from the LendingClub website: <https://www.lendingclub.com/info/download-data.action>. There are links for approved and rejected loan applications. I will be using both datasets.

## Approved Loan Dataset

This dataset has 610,917 rows and 144 columns. Each row represents a loan and the 144 columns are the various attributes used to determine if you are eligible for a loan. After cleaning the data, such as removing missing values and then applying label encoding to the categorical data, the final dataset has 87,932 rows and 42 columns. For this dataset Random Forest and XGBoost will do the predictions. The accuracy, ROC and other metrics will be evaluated to see which model is better at predictions. Kmeans and Mean Shift will cluster the top 10 features of the Random Forest model to see if there are any unknown features that can help determine loan status.

## Rejected Loan Dataset

This dataset has 19,158,655 rows and 2 columns. Each row represents a loan and the two columns are 1.) date of the loan application 2.) the number of loans for that day. This dataset was converted in to weeks. The final dataset now has 117 rows and 2 columns. SARIMA will predict 8 weeks of loans as well as forecast 12 weeks out. LSTM will also predict 8 weeks of loans

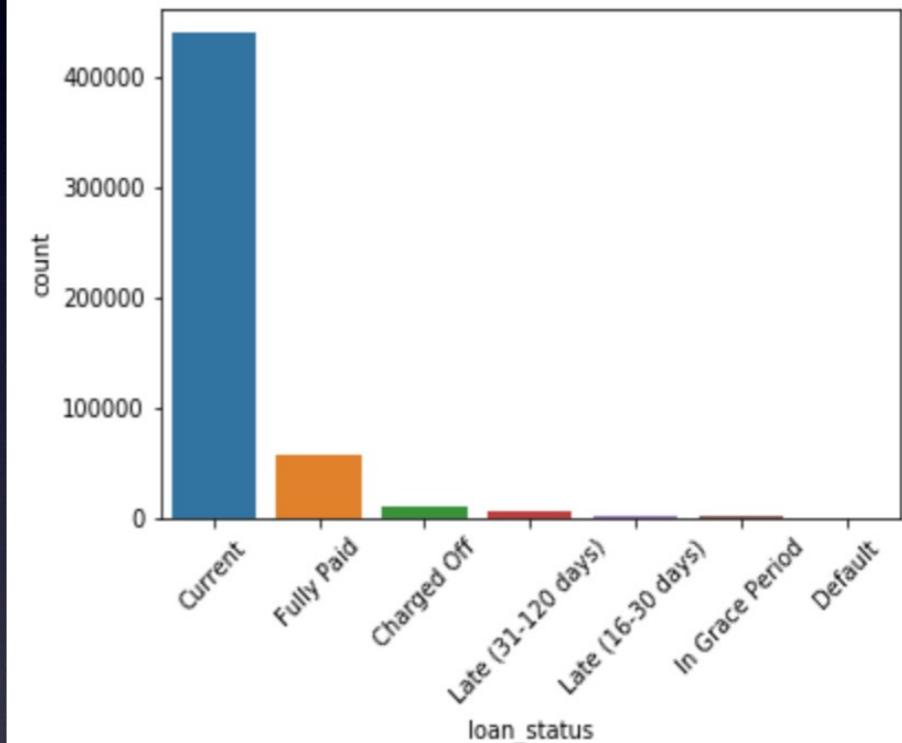
```
DatetimeIndex(['2017-01-02', '2017-01-09', '2017-01-16', '2017-01-23',
                 '2017-01-30', '2017-02-06', '2017-02-13', '2017-02-20',
                 '2017-02-27', '2017-03-06',
                 ...
                 '2019-01-21', '2019-01-28', '2019-02-04', '2019-02-11',
                 '2019-02-18', '2019-02-25', '2019-03-04', '2019-03-11',
                 '2019-03-18', '2019-03-25'],
                dtype='datetime64[ns]', name='Application_Date', length=117, freq=None)
```

# Target Variable

- A new Feature ‘bad\_loan’ was created from the categorical feature ‘loan\_status’.
- A bad loan is: Charged Off, Late(31-120 days) and Default.
- A good loan is: Fully Paid
- The other values will be discarded: Current, Late(16-30 days) and In Grace Period because they are not considered good or bad.

```
1 loan_stat = df["loan_status"]
2 sns.countplot(loan_stat)
3 stat_temp = df.loan_status.value_counts().sum()
4 plt.xticks(rotation=45)

(array([0, 1, 2, 3, 4, 5, 6]), <a list of 7 Text xtickla
```

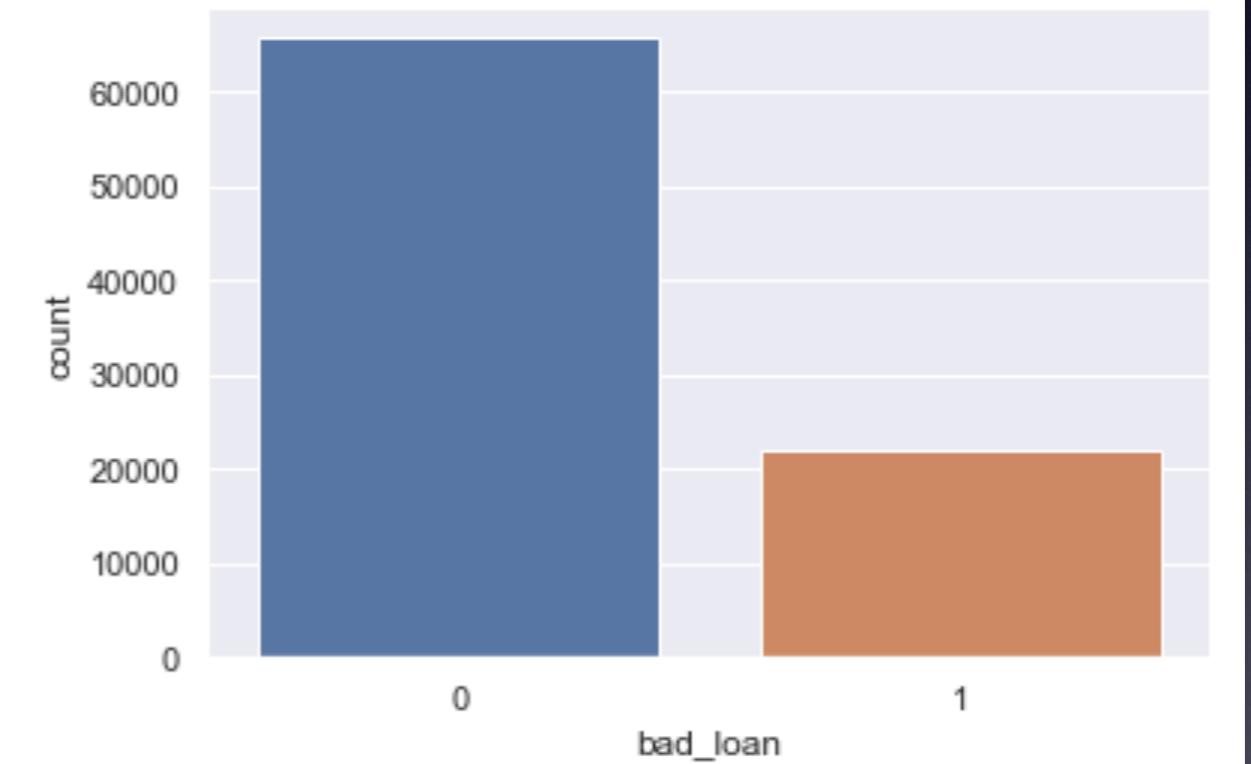


```
1 df = df[df['loan_status'] != 'Current']
2 df = df[df['loan_status'] != 'In Grace Period']
3 df = df[df['loan_status'] != 'Late (16-30 days)']
```

# Target variable

- About 74.83% of the remaining loans have been fully paid and 25% have been charged off, late or in default, so we have an unbalanced classification problem.
- If we gave all the customers loans, 25% of them would not pay back the loan.

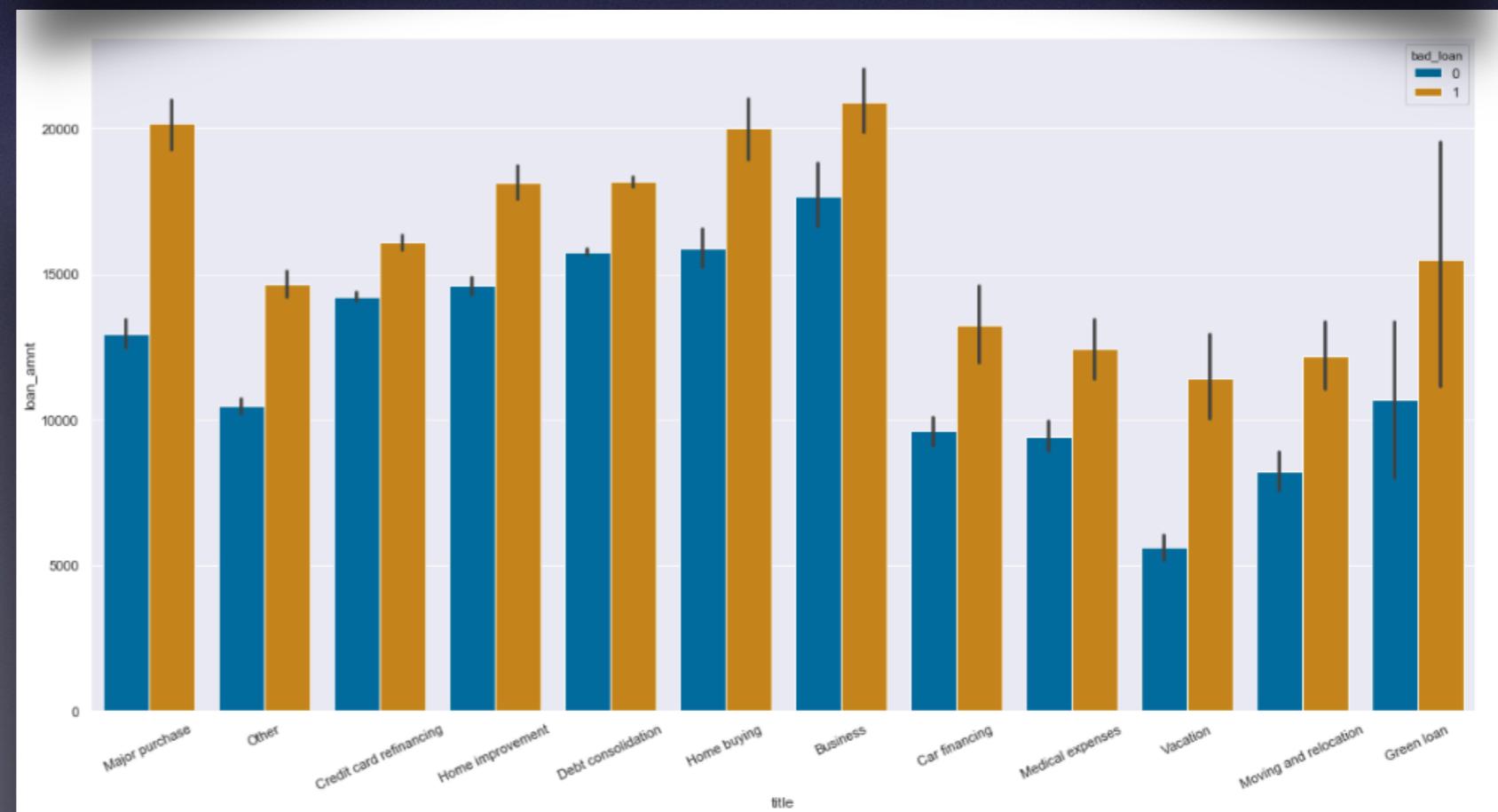
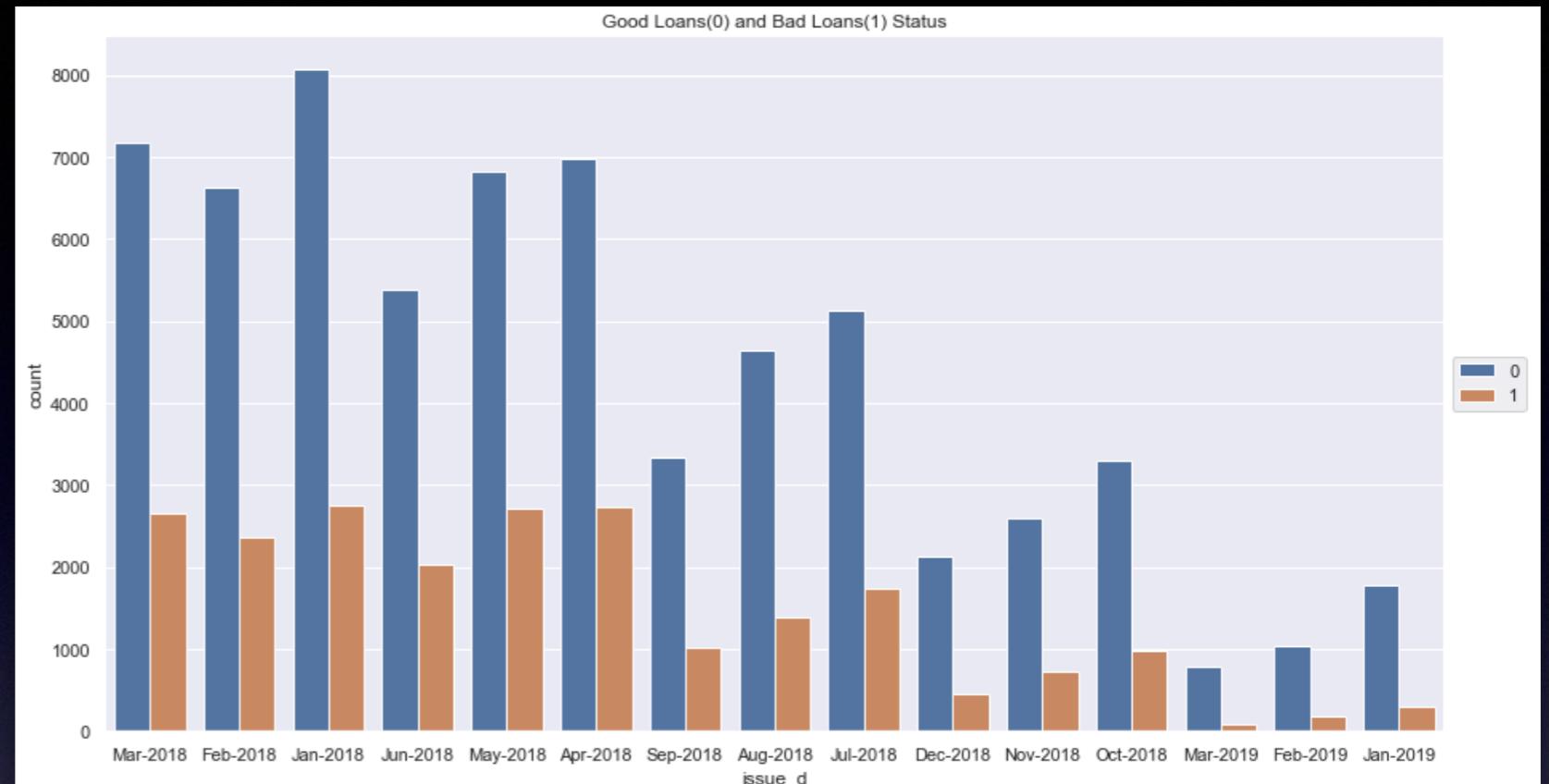
```
0      65793  
1      22130  
Name: bad_loan, dtype: int64  
baseline: 0.2517
```



## A few feature plots of good(0) vs bad loans(1)

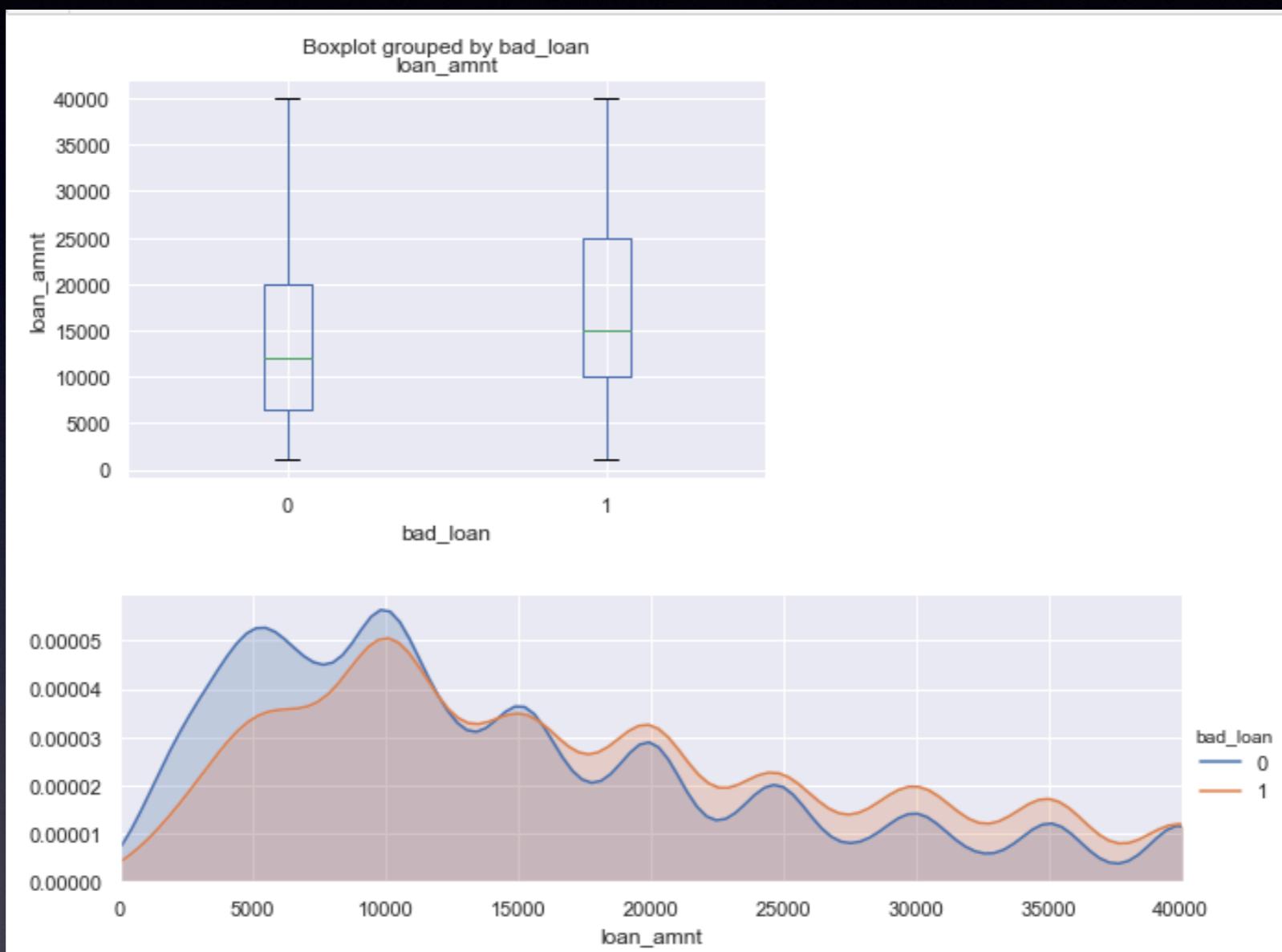
Months January - June have the most defaulted loans.

Business loans are the top defaulters and they have the highest loan amounts.



## good(0) vs bad loans(1)

On average, customers who default on a loan have a higher loan amount.



# Supervised Learning

Random Forest, ADABoost Random Forest and XGBoost are the algorithms used to determine the percentage of good loans that will be issued. Grid-search was used to find the best hyper-parameters for each model. Below is the result.

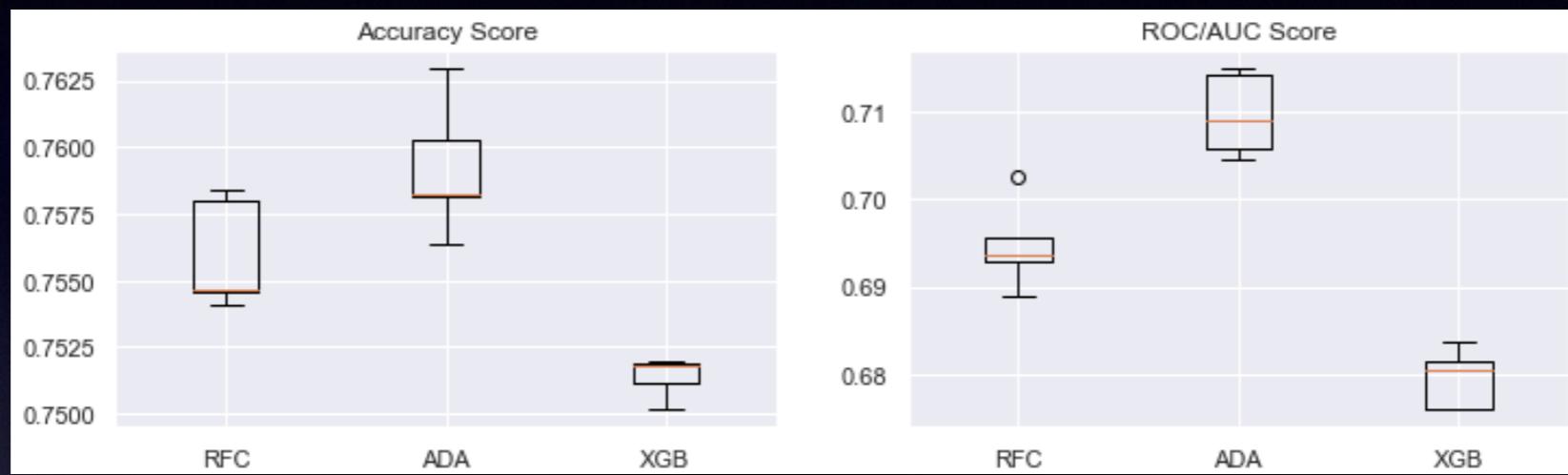
```
rf_final = RandomForestClassifier(max_depth=8,  
                                 min_samples_leaf=10,  
                                 min_samples_split=60,  
                                 max_features=50,  
                                 n_estimators=20)
```

```
ada_final = AdaBoostClassifier(base_estimator=rf_final,  
                               n_estimators=14)
```

```
xgb_final = XGBClassifier(max_depth=6,  
                           n_estimators=8,  
                           learning_rate=0.0125,  
                           subsample=0.03,  
                           colsample_bytree=0.05,  
                           colsample_bylevel=0.005)
```

# Results

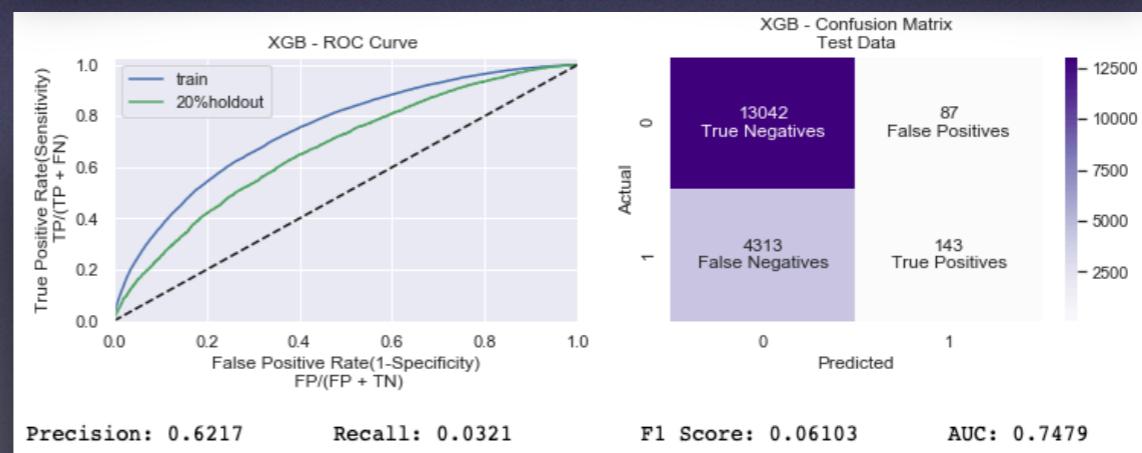
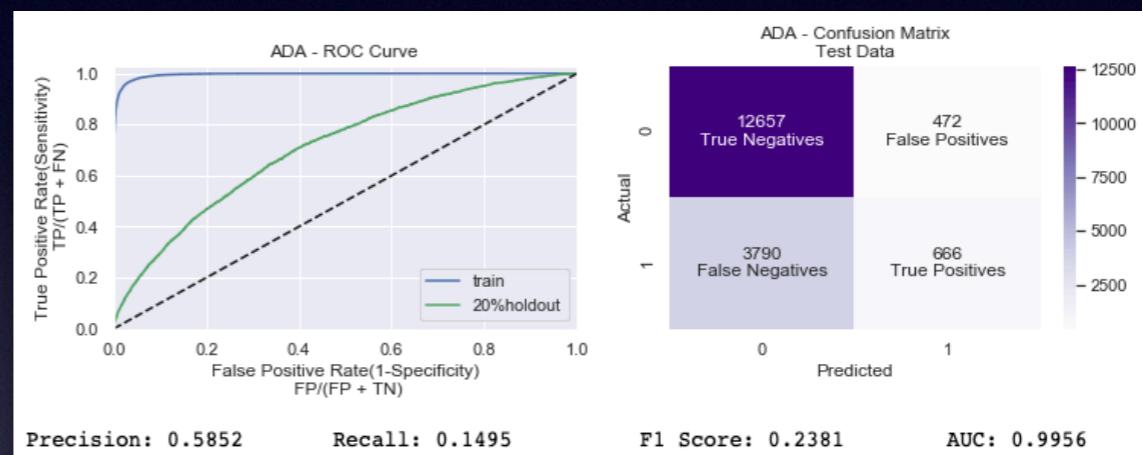
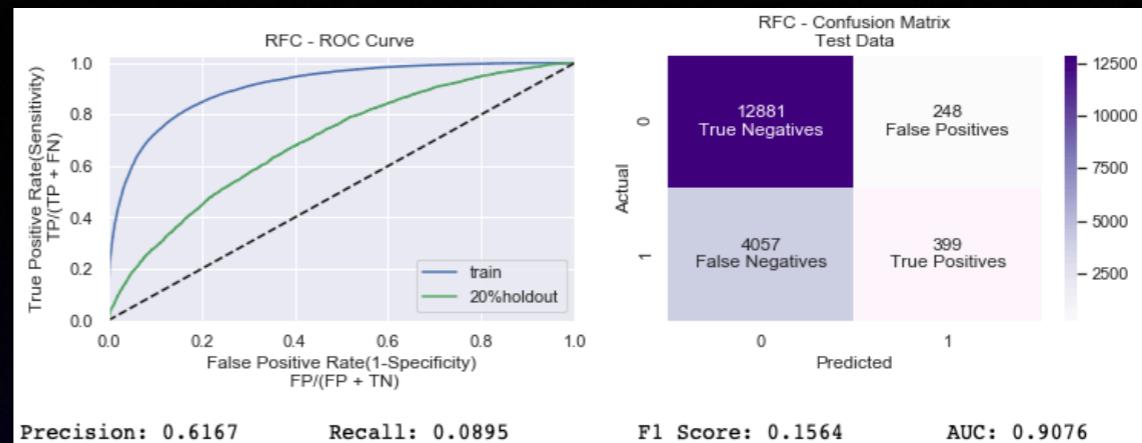
Random Forest, ADABoost Random Forest and XGBoost accuracy scores are all approximately 0.75. XGB is the most stable and ADABoost has the better AUC score.



	Accuracy		ROC	
	score	(std)	score	(std)
RFC	0.7559	(0.0019)	0.6947	(0.0045)
ADA	0.7592	(0.0022)	0.7096	(0.0043)
XGB	0.7514	(0.0007)	0.6795	(0.0030)
	baseline 0.7483			

target variable bad loans=1

- Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answers is of all loans that are labeled as bad, how many actually were bad? All models have a ~60% precision.
- Recall is the ratio of correctly predicted positive observations to the all observations. The question recall answers is: Of all the loans that are truly bad, how many did the model label correctly? The recall for all the models is very low.
- To make sure LendingClub doesn't lose lots of money, customers who don't pay back their loan, we need to decrease the number of False Negatives.
- By decreasing the False Negatives (loans predicted as being paid back that are actually not paid) the recall score will increase.



# XGBoost weighted model

- In an attempt to increase the Recall score, the parameter `scale_pos_weight` was added to XGBoost.
- The parameter `scale_pos_weight` controls the balance of positive and negative weights which is useful for unbalanced classes.
- There will be a trade off in the accuracy score when the weight is applied. The accuracy will decrease but it is better to have less borrowers who don't pay back the loan than a high accuracy score for this model.
- The weight of 5.5 was chosen as the 'best'. It decreased the False Negatives from 4313 to 1749, increased the False Positives.

```
weights = [1, 3.5, 4, 4.5 ,5, 5.5, 6, 6.5, 14]
for x in weights:
    xgb_bal = XGBClassifier(max_depth=12,
                           n_estimators=20,
                           learning_rate=0.03,
                           subsample=0.1,
                           colsample_bytree=0.5,
                           colsample_bylevel=0.5,
                           scale_pos_weight=x)
    xgb_bal.fit(X_train, y_train)
    predictions = xgb_bal.predict(X_test)
```

## Weight of 1

Recall Score: 0.03  
Accuracy Score: 0.75  
Confusion Matrix  
[[13042 87]  
[ 4313 143]]

## Weight of 5.5

Recall Score: 0.61  
Accuracy Score: 0.62  
Confusion Matrix  
[[8151 4978]  
[ 1749 2707]]

## Weight of 3.5

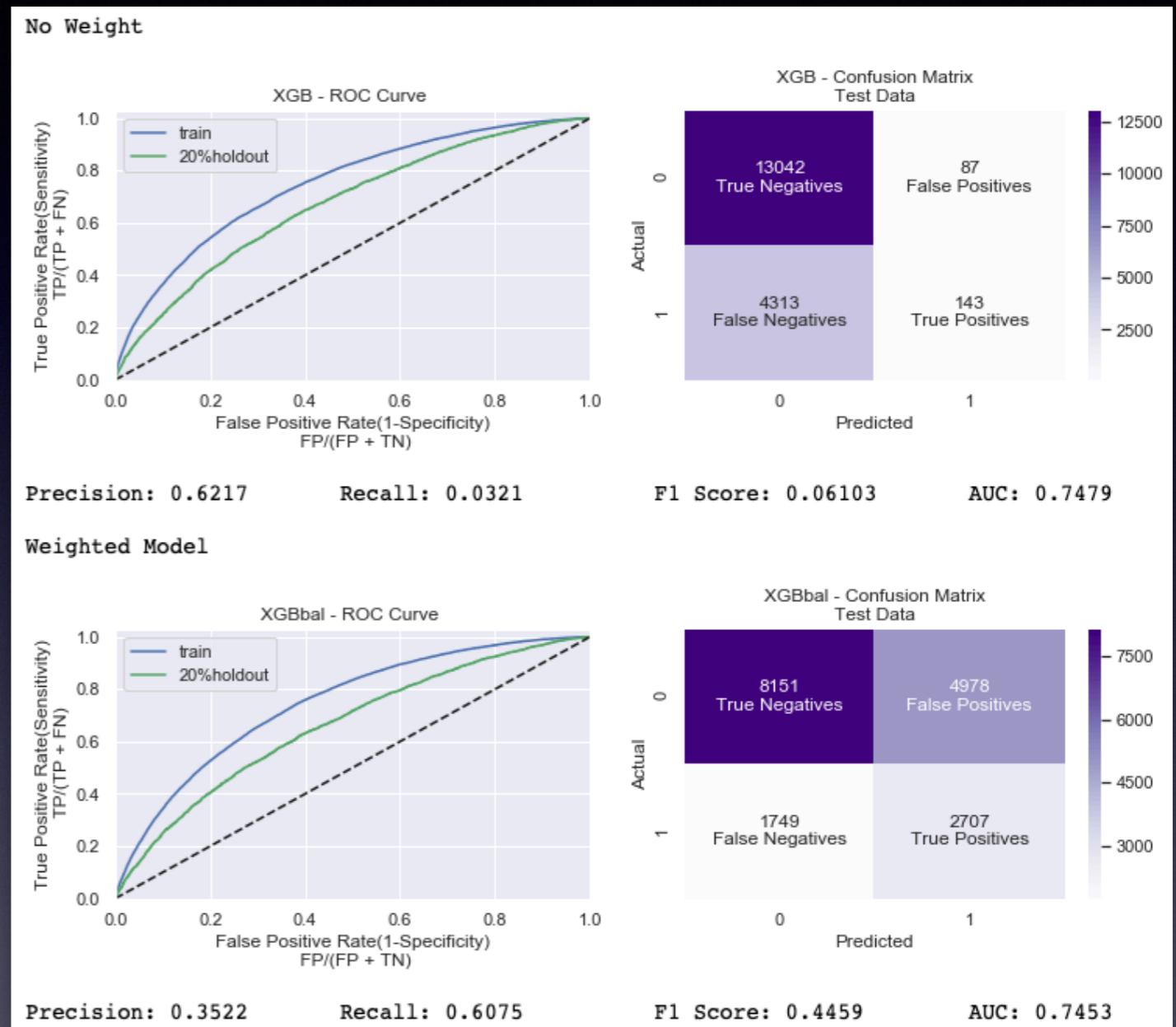
Recall Score: 0.43  
Accuracy Score: 0.70  
Confusion Matrix  
[[10379 2750]  
[ 2541 1915]]

## Weight of 14

Recall Score: 0.91  
Accuracy Score: 0.39  
Confusion Matrix  
[[ 2884 10245]  
[ 417 4039]]

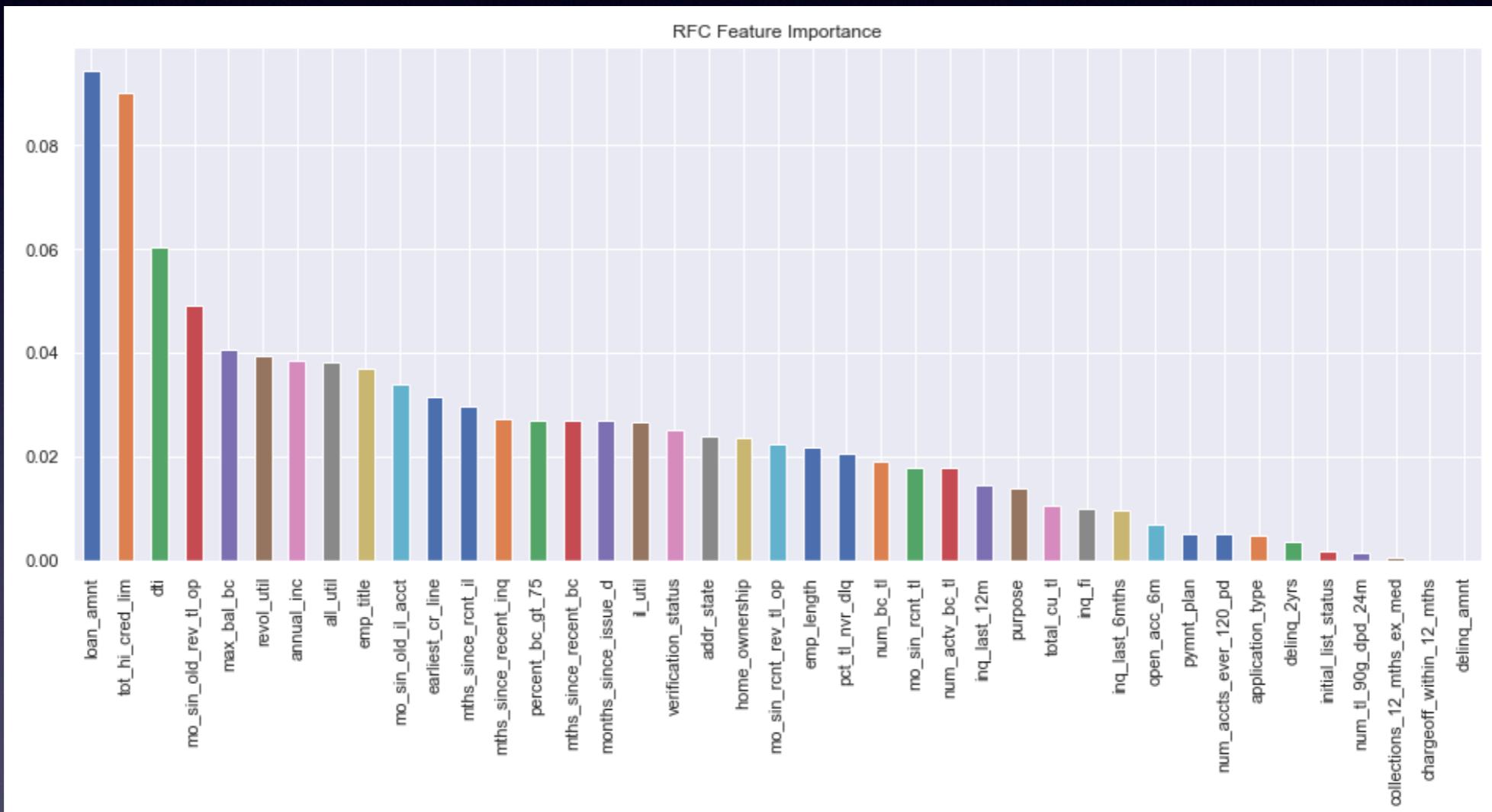
# XGBoost with and without weight

- The Recall increased by 18.93%, F1 score increased by 7.3% and Precision decrease by 56.65% as expected.
- The False Negatives decrease from 4313 to 1749. So instead of 4313 loans as a LendingClub write off only 1749 loans will be written off.



# Feature Importance

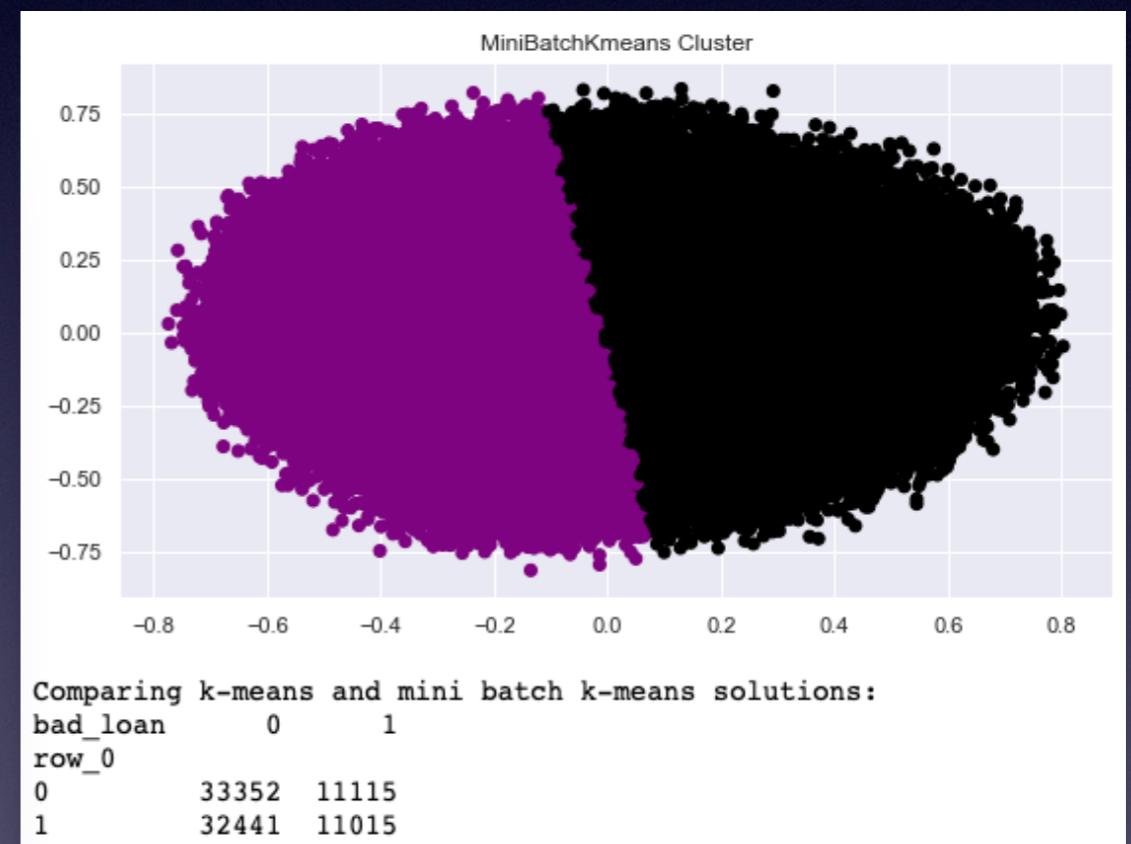
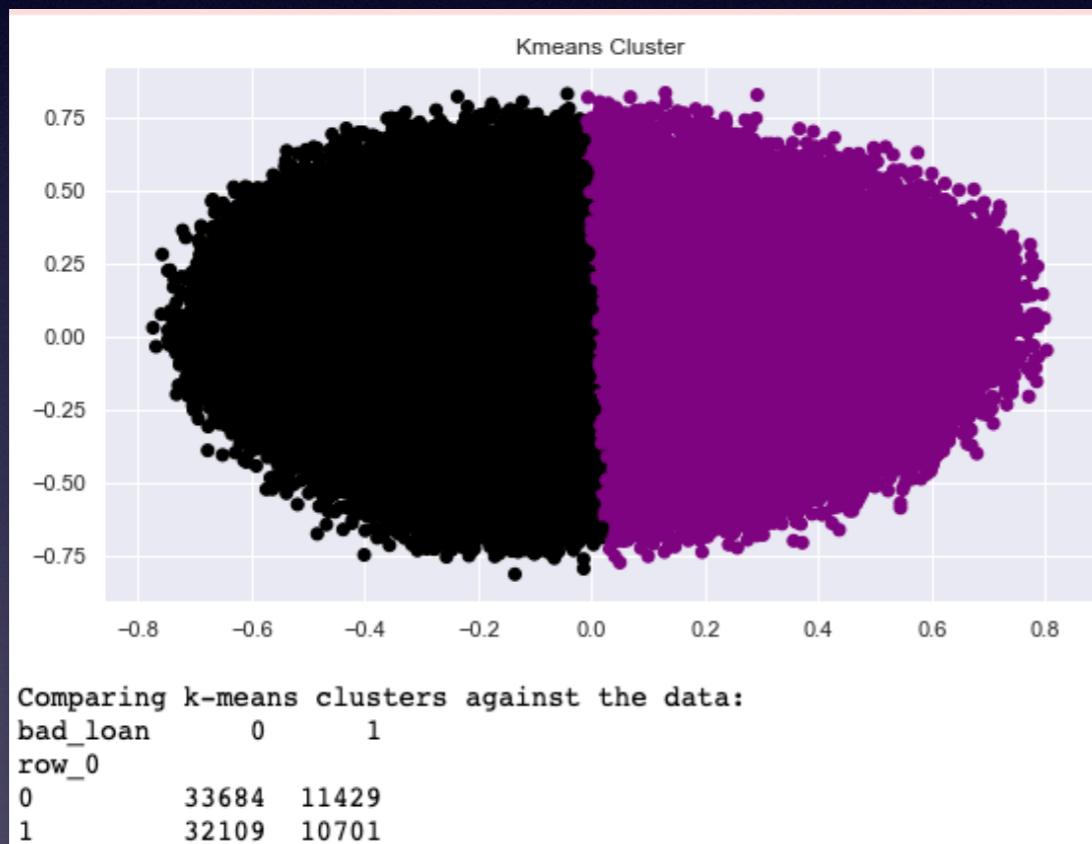
The top 10 features will be used for Kmeans & Mean Shift clustering.



# KMeans & KMeans Minibatch

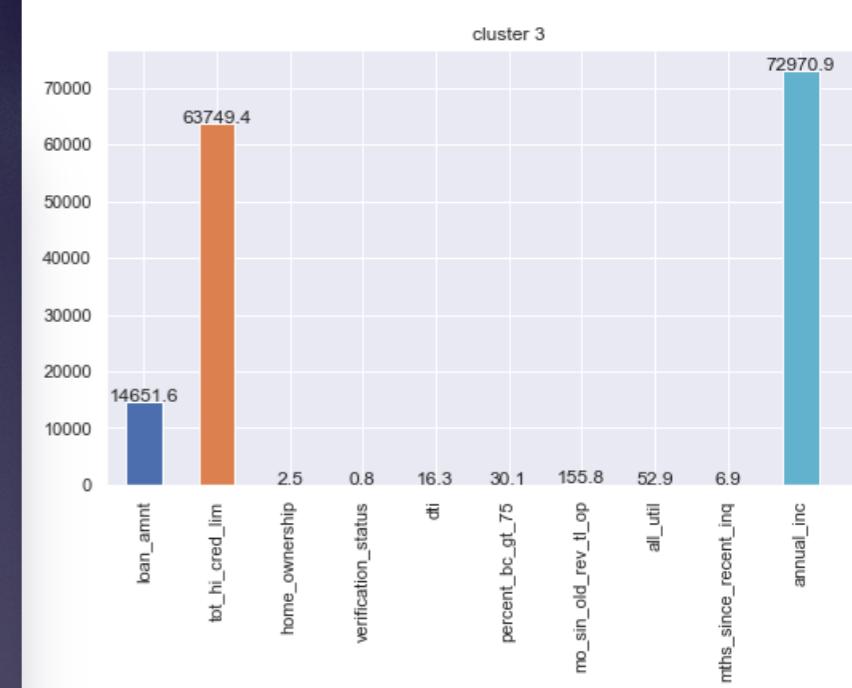
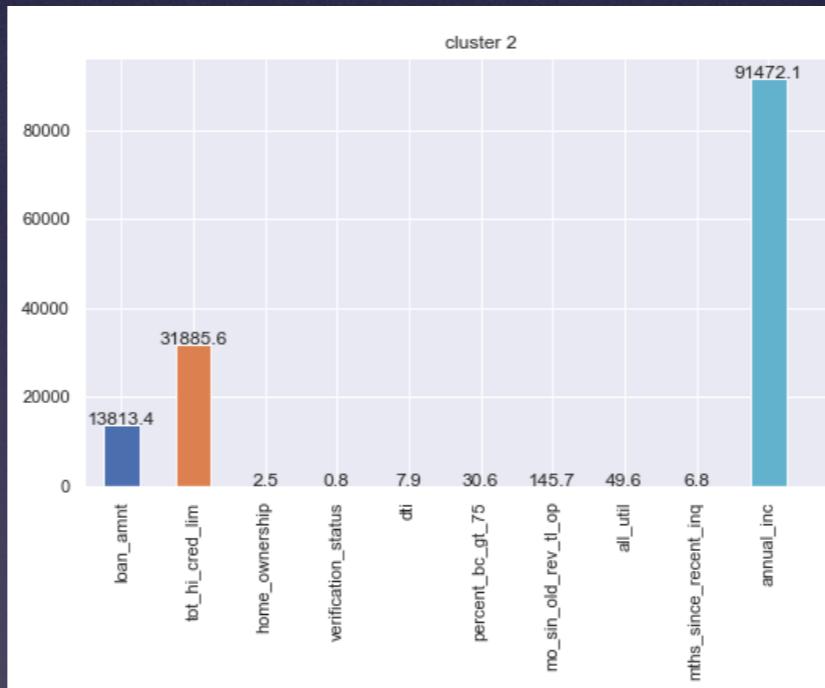
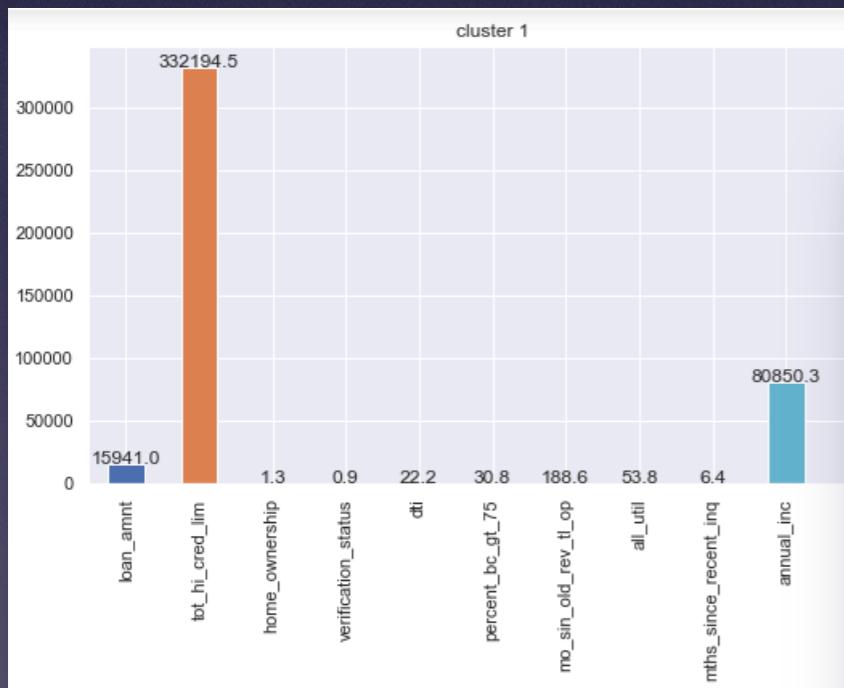
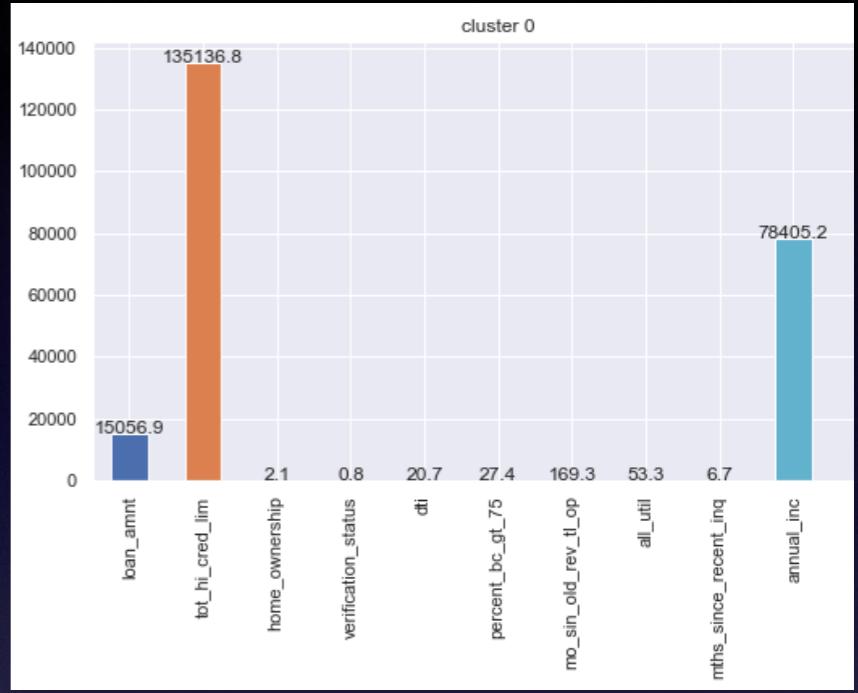
44,298 (10701+33684) correctly classified  
43,625 (32109+11429) mis-classified

43,207 (11015+33352) correctly classified  
44,716 (32441+11115) mis-classified



# Kmeans Clusters on RFC Features

- Cluster 0 - people who have **high credit limit of 135k, annual income of 78k** and 15k loan amount
- Cluster 1 - people who have **high credit limit of 332k** and an **annual income of 80k**.
- Cluster 2 - people **annual income 91k, high credit limit of 32k** and loan amount 14k.
- Cluster 3 -**annual income 73k, high credit limit 32k** and loan amount of 14k
- So for potential customers who have a high credit limit that is over 30% of their income they will most likely default on the loan.



# Time Series

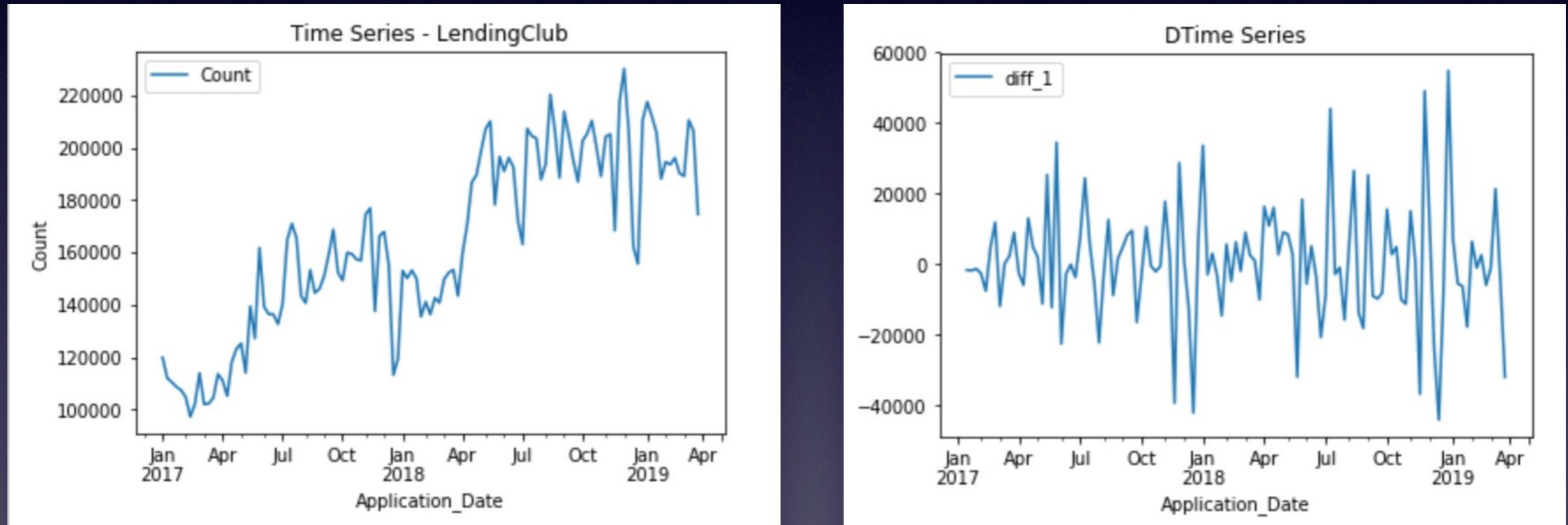
- For time series we are using the rejected loan application dataset. The original dataset has 19158655 rows but after converting the date to weeks, it is reduced to 117 rows.
- SARIMA will predict 8 weeks of rejections as well as forecast rejection 12 weeks out. LSTM will also predict 8 weeks of loans using the same data.

## Rejected Loans (weekly)

Application_Date	Count
2017-01-02	119903
2017-01-09	112116
2017-01-16	110457
2017-01-23	108717
2017-01-30	107435

# Original and Differenced time series

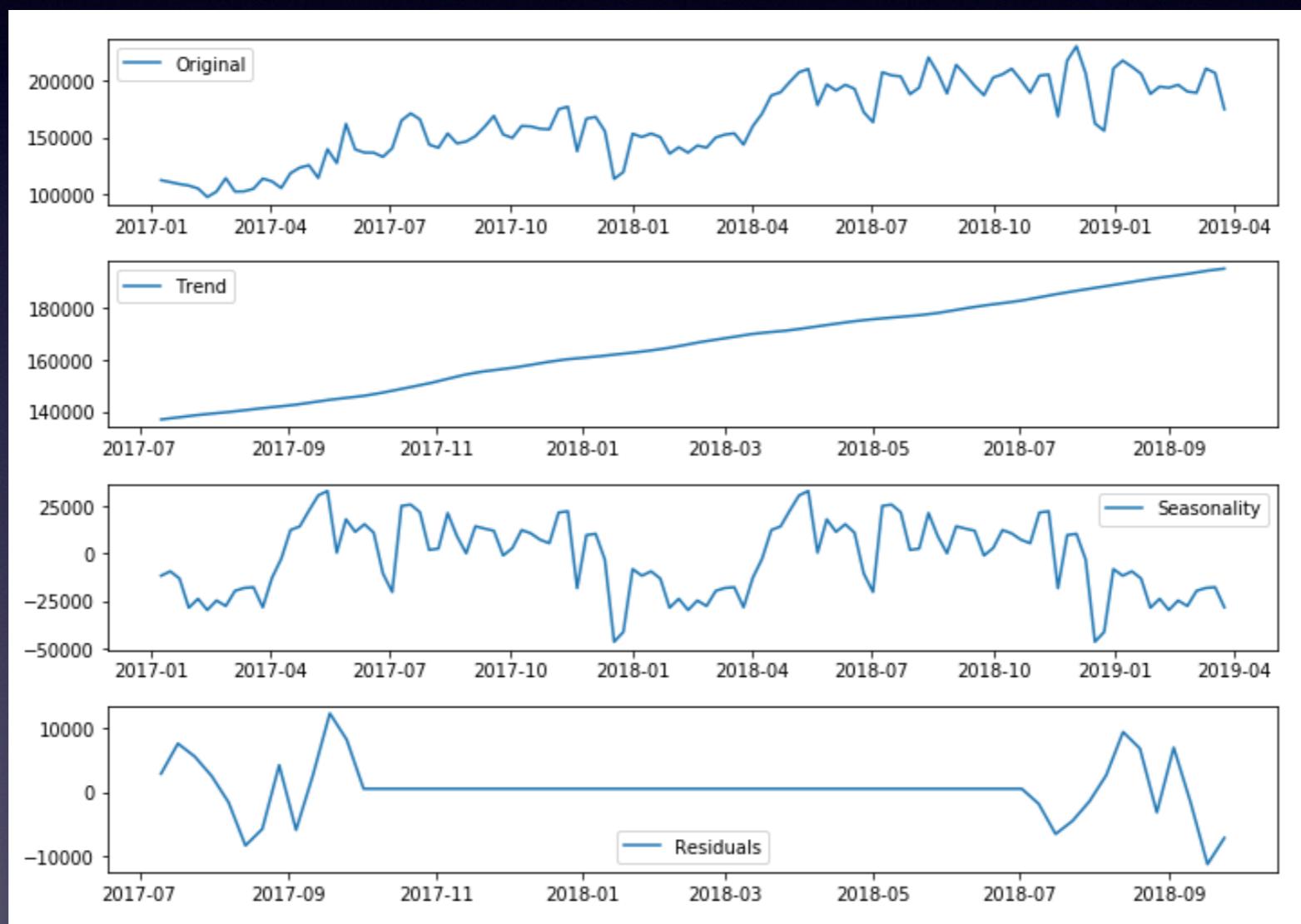
- The data is not stationary therefore a difference of one is done to make it stationary.



# Decompose Seasonality

A statistical task that deconstructs a time series into several components.

Here we can see that the trend and seasonality are separated out from data and we can model the residuals.



# Augmented Dickey-Fuller test

The null hypothesis for this test is that there is a unit root.  
The basic alternate is that the time series is stationary.

A large p-value ( $> 0.05$ ) indicates weak evidence against the null hypothesis, so you fail to reject the null hypothesis. The p-value is greater than 0.05, this is an indication that the model is not stationary and we have to take a difference to make it stationary.

```
raw data
Test Statistic -1.653998
p-value 0.454956
#lags used 3.000000
#nobs used 112.000000

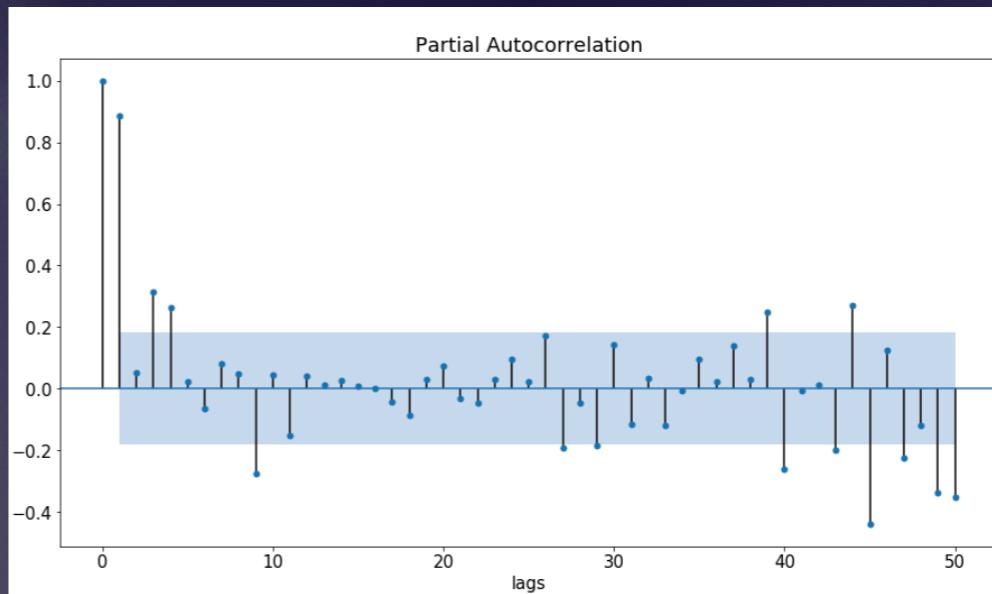
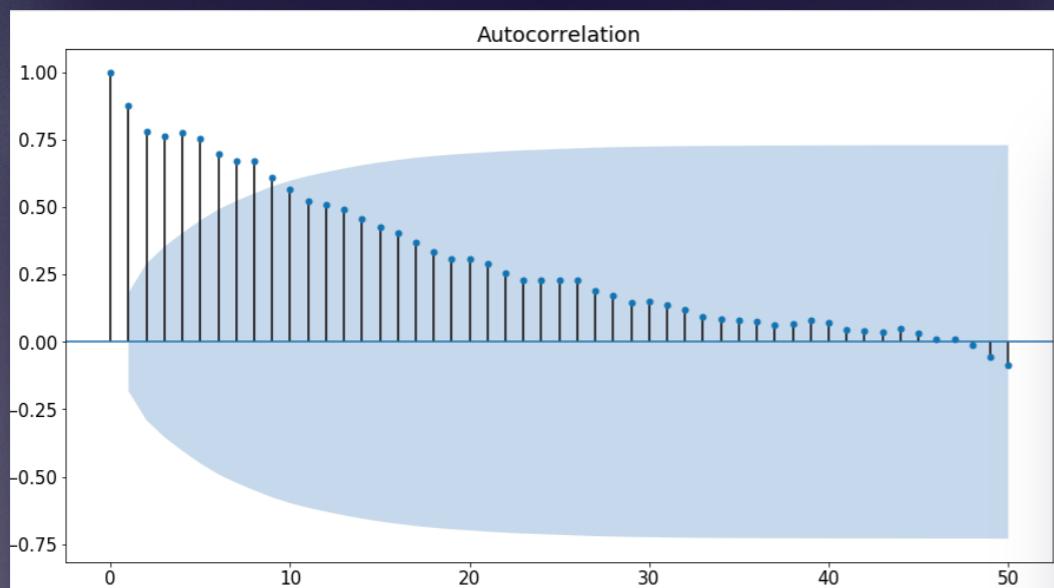
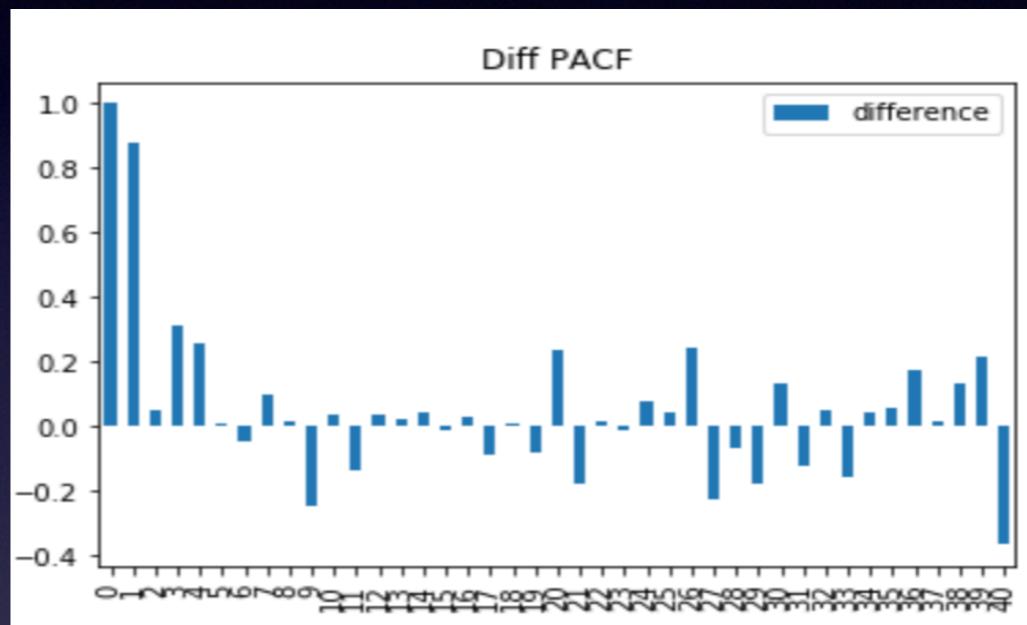
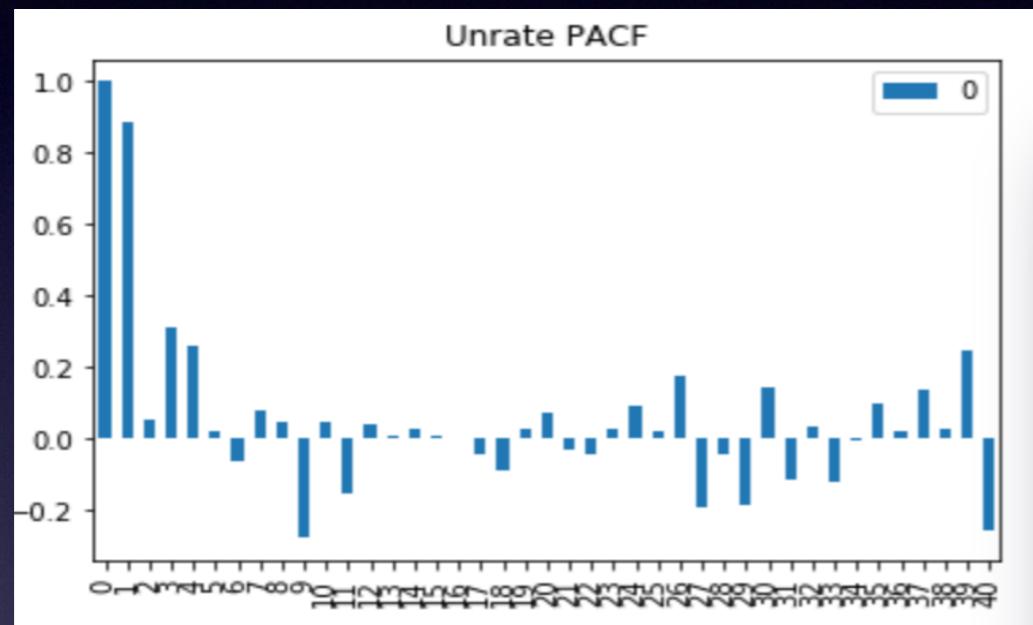
1% -3.4901313156261384
5% -2.8877122815688776
10% -2.5807296460459184
```

# PACF original and differenced time series

The initial and the differenced plots both have a strong autocorrelation at 1.

The Autocorrelation plot says the value in time 't' has some correlation with the previous.  
The partial autocorrelation plot ask's the question 'how many lags do you need?'

A customer's annual income for 2019 may have a relationship with their income from 2012 but we only need the annual income from 2018 to be accurate.



# SARIMA Time Series model results

```
stepwise_model =  
auto_arima(dflIndex['Rejected_Loan_Count'],  
           start_p=0, start_q=0,  
           max_p=13, max_q=13, m=13,  
           start_P=0, start_Q=0,  
           seasonal=True,  
           d=1, D=1, trace=True,  
           error_action='ignore',  
           suppress_warnings=True,  
           random_state=42,  
           n_fits=3,  
           stepwise=True)  
  
stepwise_model.summary()
```

SARIMAX(0, 1, 2)x(1, 1, 1, 13)

This was the best fit, the lowest AIC. The seasonality is 13 which is ok as the data is quarterly.

Statespace Model Results						
Dep. Variable:	y	No. Observations:	116			
Model:	SARIMAX(0, 1, 2)x(1, 1, 1, 13)	Log Likelihood	-1134.857			
Date:	Tue, 16 Jul 2019	AIC	2281.713			
Time:	16:18:21	BIC	2297.463			
Sample:	0	HQIC	2288.091			
	- 116					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
intercept	-315.9325	725.235	-0.436	0.663	-1737.367	1105.502
ma.L1	-0.3026	0.238	-1.272	0.204	-0.769	0.164
ma.L2	-0.2153	0.198	-1.086	0.278	-0.604	0.173
ar.S.L13	-0.2300	0.210	-1.096	0.273	-0.641	0.181
ma.S.L13	-0.6329	0.261	-2.428	0.015	-1.144	-0.122
sigma2	4.291e+08	0.001	7.23e+11	0.000	4.29e+08	4.29e+08
Ljung-Box (Q):	38.70	Jarque-Bera (JB):	0.85			
Prob(Q):	0.53	Prob(JB):	0.65			
Heteroskedasticity (H):	1.30	Skew:	0.14			
Prob(H) (two-sided):	0.45	Kurtosis:	3.34			

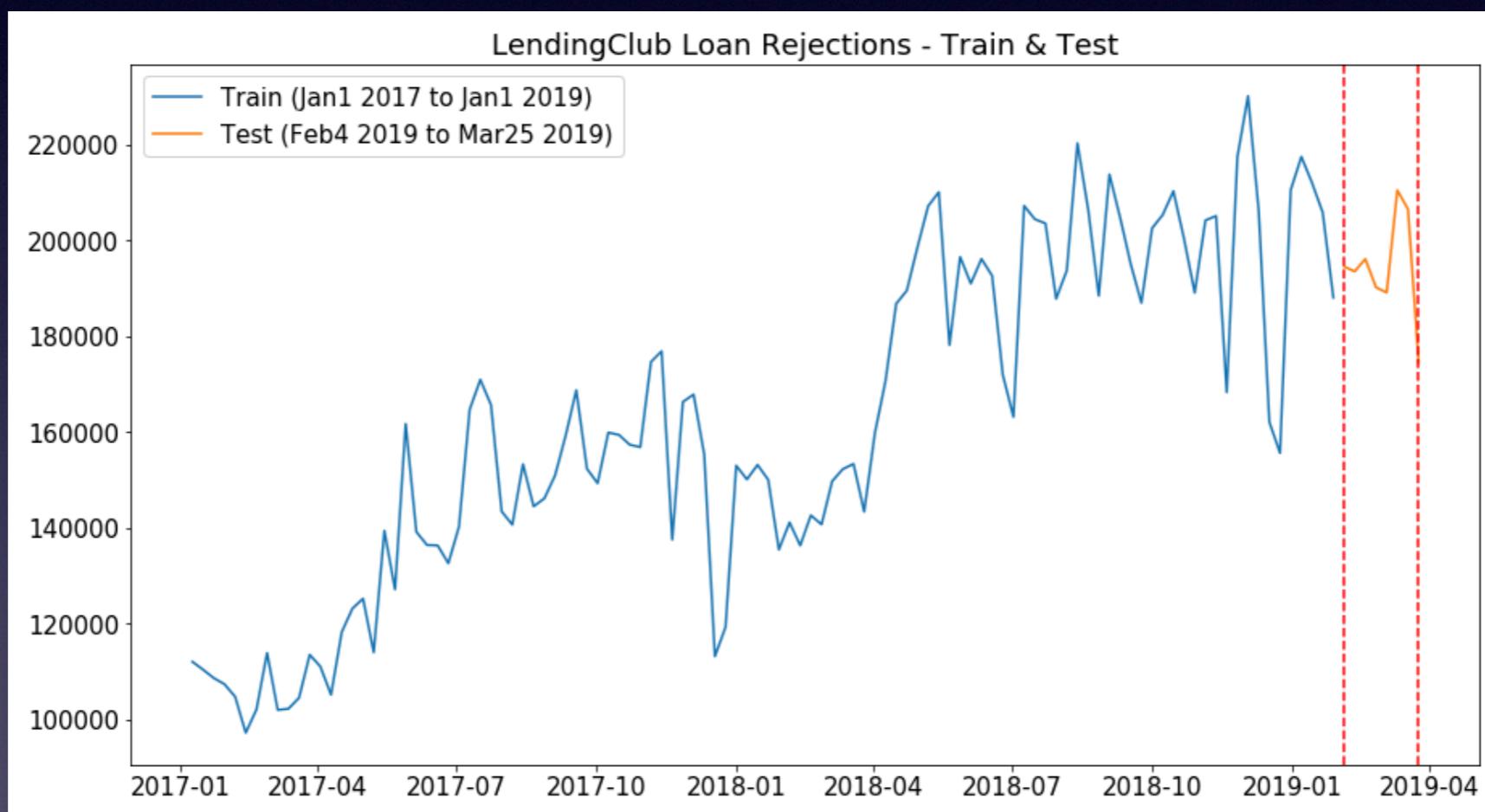
# Forecast of 8 weeks

Loan dataset will have 116 rows.

train, test = dfIndex[:108], dfIndex[108:]

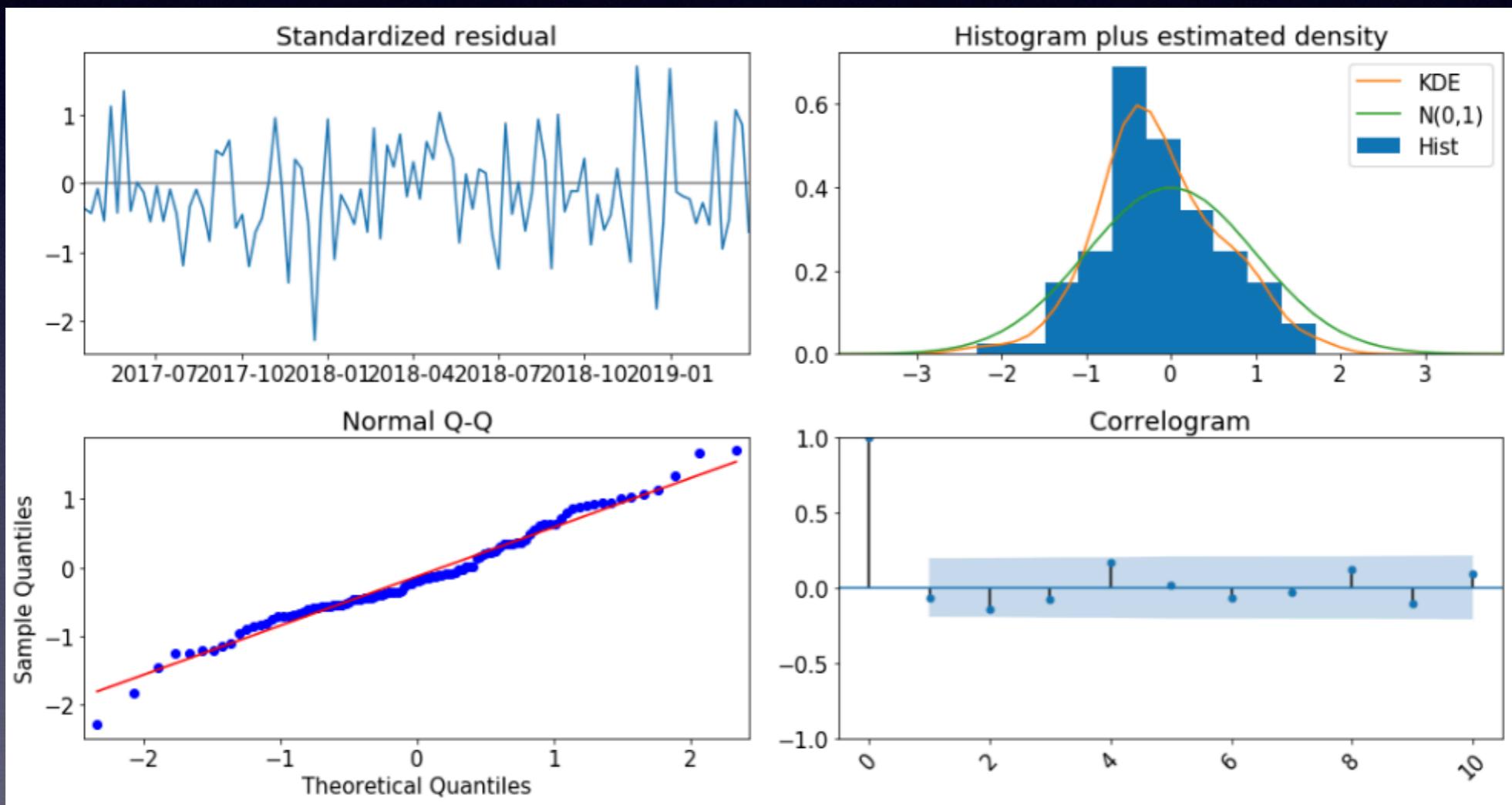
Train: 108 rows weekly loan rejections from Jan1 2017 to Jan1 2019

Test: 8 rows of weekly loan rejections from Feb4 2019 to Mar25 2019



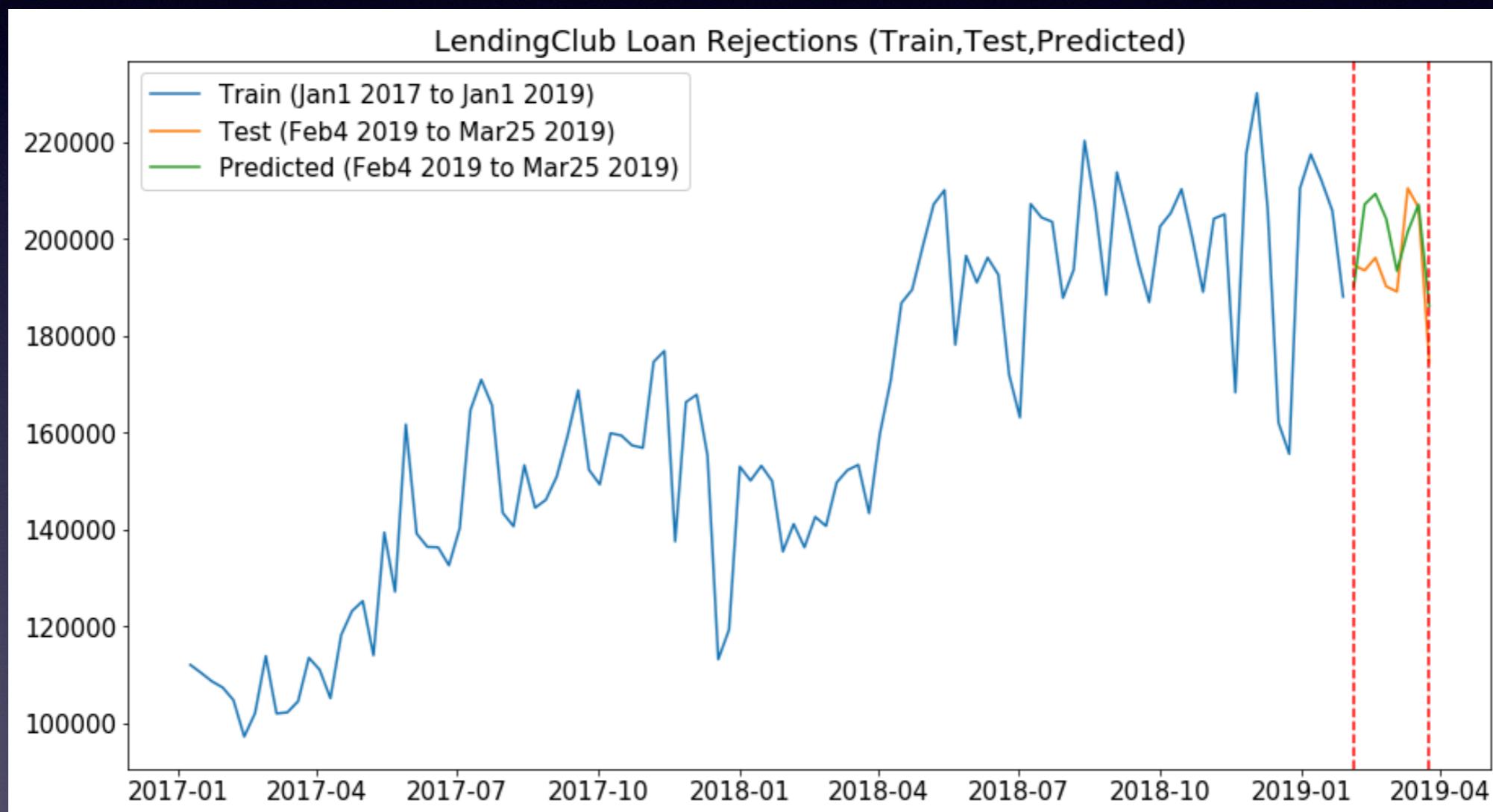
# Results

- The residuals look random therefore the model doesn't seem to have any bias.
- The histogram distribution has a Gaussian look.
- The Q-Q plot also shows a normal distribution but with a few outliers.
- There isn't any obvious autocorrelation trend in the plot.



# SARIMA Forecast

- The actual and predicted forecast comparison plots.
- For the first 4 weeks the predicted is higher than the actual but after that the predictions are close to the actual.



# SARIMA Forecast Errors

Application_Date	Loan_Count	Predicted_Count	Error	Predictions
2019-02-04	194570	190282.51	4287.49	200641.32
2019-02-11	193545	207165.14	-13620.14	212867.37
2019-02-18	196149	209330.69	-13181.69	191871.74
2019-02-25	190241	204186.00	-13945.00	202893.15
2019-03-04	189141	193419.03	-4278.03	213443.53
2019-03-11	210487	201513.06	8973.94	206000.16
2019-03-18	206549	207129.72	-580.72	190300.88
2019-03-25	174618	186142.10	-11524.10	181063.40

Mean of Number of Loans: 194,412.50

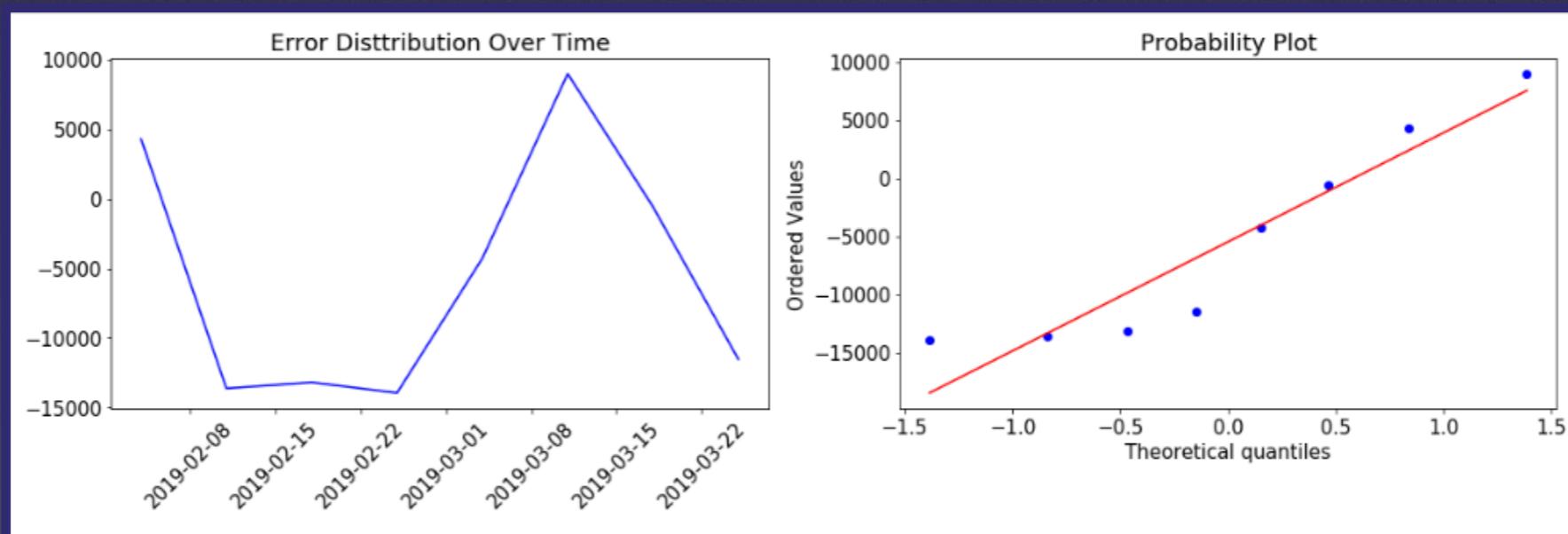
Root Mean Square Error: 10,025.50

Mean Squared Error: 100,510,732.85

Percent Mean Error: 5,483.53

The model is off by 5.16%.

The with the mean number of loan = 194,412.50 and the root mean square error of 10,025.50.



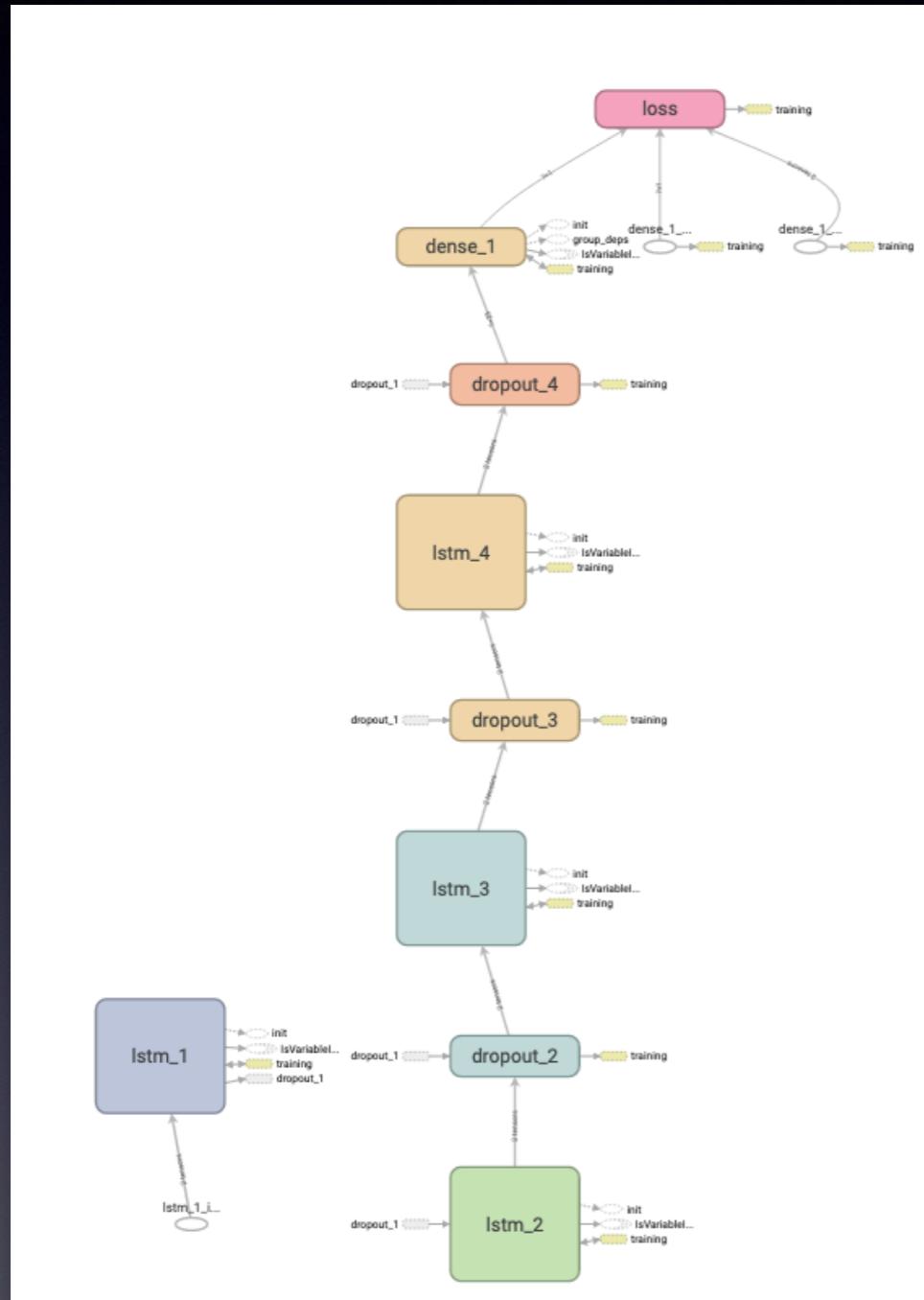
# LSTM Time Series

- Long short-term memory(LSTM) is capable of learning long-term dependencies by remembering information for long periods of time.
- It also uses a sigmoid function which helps determine what to keep( $>0$ ) or discard( $<0$ ).
- LSTM will look at the previous 8 weeks(time steps) to predict the 9th week.
- After many different combinations of the parameters, this yielded the best result:
  - units = 100
  - dropout = 0.2
  - epochs = 900
  - batch size = 1
  - loss = mean squared error

# LSTM Sequential Model & Summary

Sequential LSTM was chosen because it allows the preservation of gradients. The memory cell remembers the first input as long as the forget gate is open and the input gate is closed. The output gate provides finer control to switch the output layer on or off without altering the cell contents.

The model has 3 layers

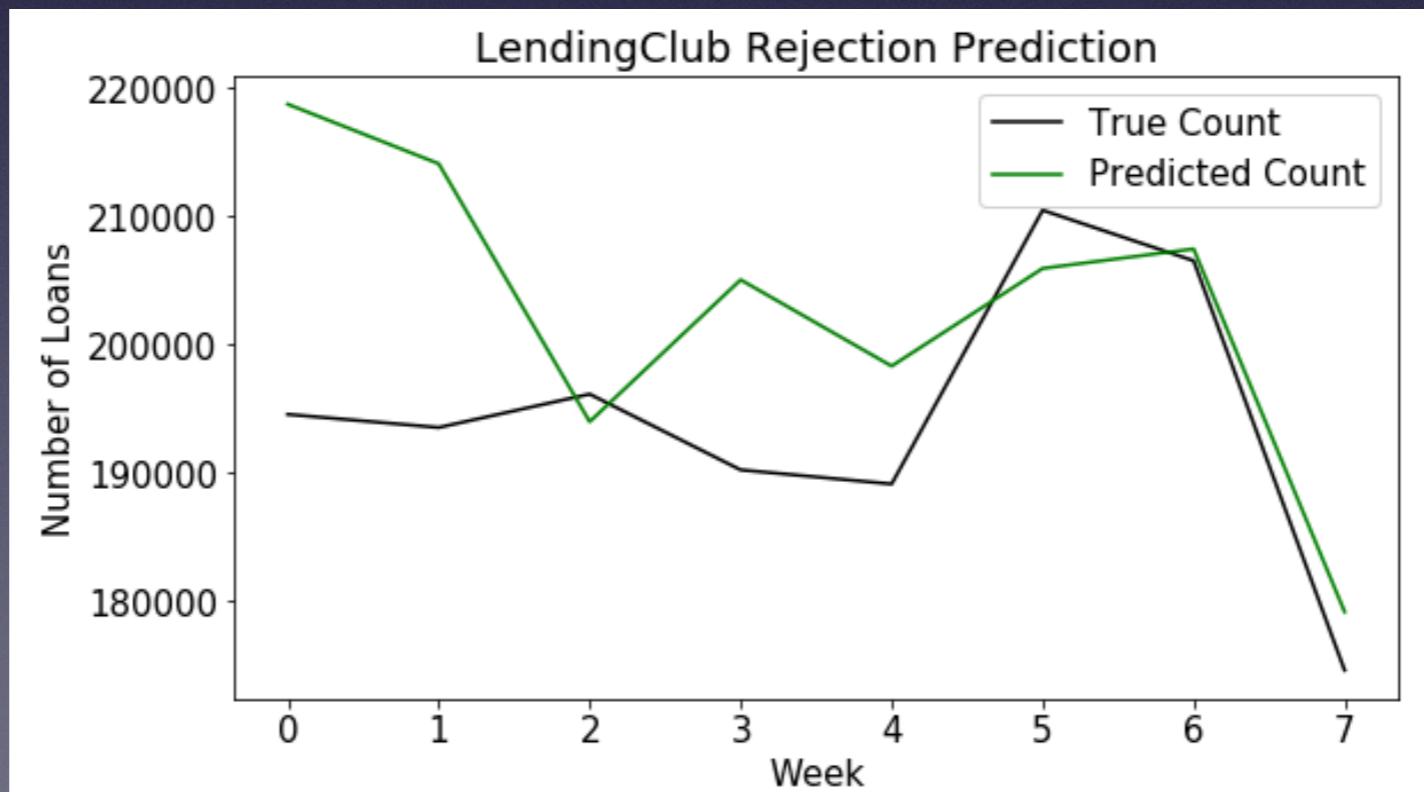


```
Istm_model = Sequential()  
Istm_model.add(LSTM(units=100,return_sequences=True, input_shape=(X_train.shape[1], 1)))  
Istm_model.add(Dropout(0.2))  
Istm_model.add(LSTM(units=100, return_sequences=True))  
Istm_model.add(Dropout(0.2))  
Istm_model.add(LSTM(units=100, return_sequences=True))  
Istm_model.add(Dropout(0.2))  
Istm_model.add(LSTM(units=100))  
Istm_model.add(Dropout(0.2))  
Istm_model.add(Dense(units=1))  
Istm_model.compile(optimizer='adam', loss='mean_squared_error')  
history = Istm_model.fit(X_train, y_train, epochs=900,batch_size=1,callbacks=[tensorboard])
```

Layer(type)	Output Shape	Param#
Istm_29 (LSTM)	(None,8 (Dense))	40800
dropout_29 (Dropout)	(None,8 (Dense))	0
Istm_30 (LSTM)	(None,8 (Dense))	80400
dropout_30 (Dropout)	(None,8 (Dense))	0
Istm_31 (LSTM)	(None,8 (Dense))	80400
dropout_31 (Dropout)	(None,8 (Dense))	0
Istm_32 (LSTM)	(None,100)	80400
dropout_32 (Dropout)	(None,100)	0
dense_8 (Dense)	(None,1)	101
Total params: 282,101		
Trainable params: 282,101		
Non-trainable params: 0		

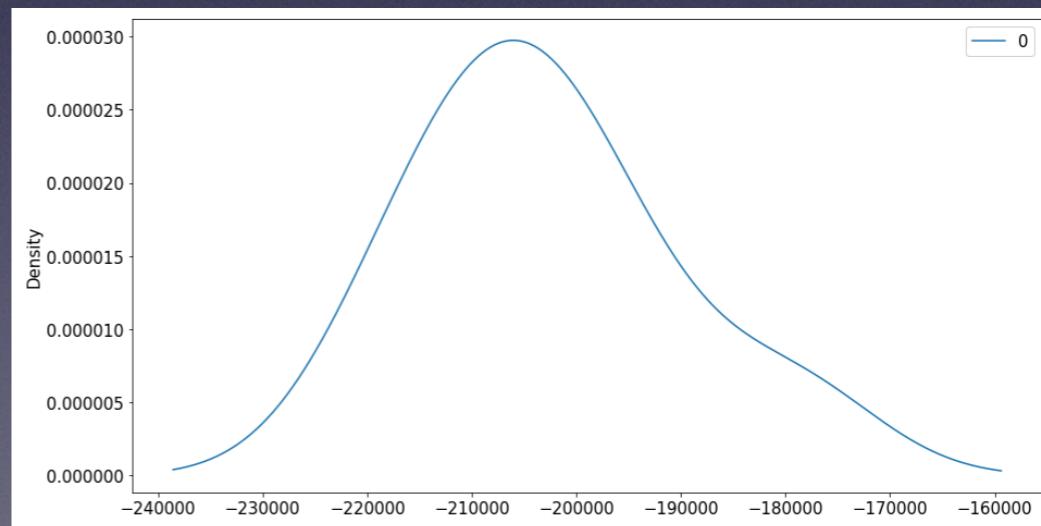
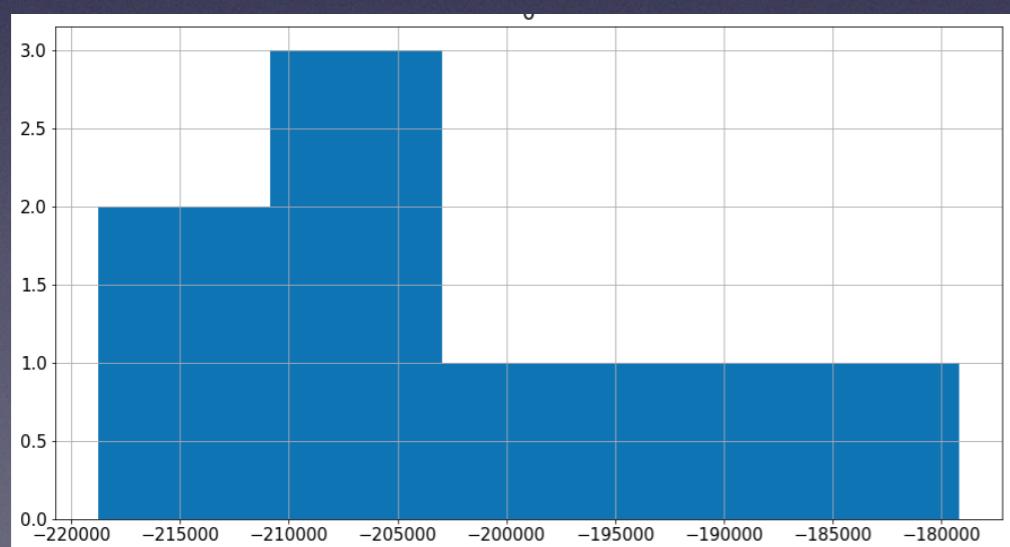
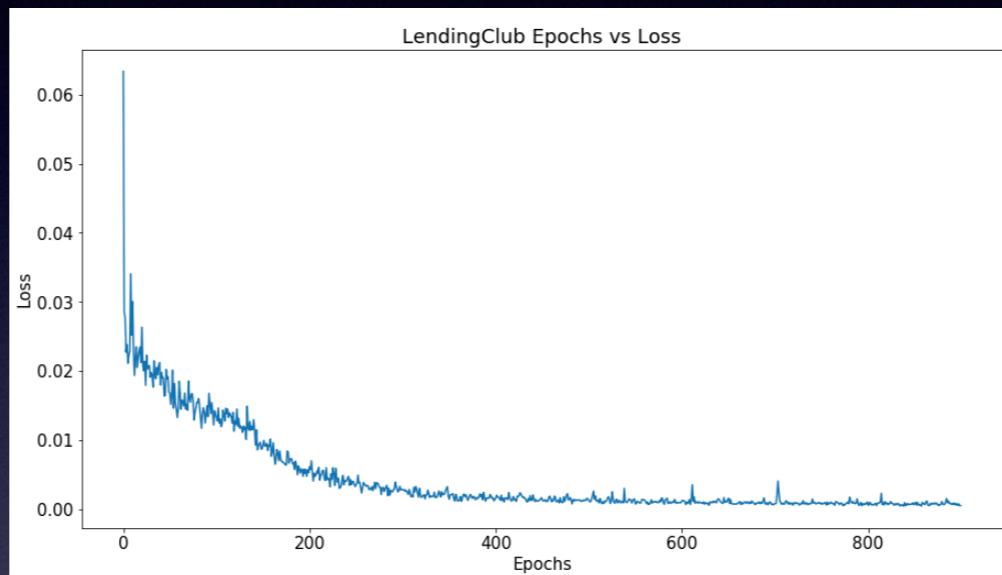
# LSTM Prediction Result

- The first 5 months the predicted number of rejected loans is a lot greater than the actual number of loans.
- The 5th thru the 8th month predictions were a lot better.
- RMSE Test Score: 13,033.17
- MSE Test Score: 169,863,469.53
- The model is off by 6.70%.
- The with the mean number of loan = 194,412.50
- and the root mean square error of 13,033.17
- SARIMA predicted the 8 weeks of loans better than LSTM by 1.54%.



# Loss, Residuals and Density Plots

- There is a sharp drop off for the loss at the start and then it tries to level out
- For the residual plot, the distribution does look Gaussian with right skewness (the mean is to the right of the median)



# Summary

- The supervised learning models predicted too many False Negatives (applications which were predicted as good but they were actually bad loans). Using a weighted XGB model decrease the False Positives which therefore increased the recall score from 3% to 60% but the penalty was the accuracy score decreased from 50% o 62%.
- For future work, it is worth testing the model using different sampling(up/down) methods to balance the target variable.
- The top 10 features from Random Forest were analyzed with KMeans and Mean Shift. Kmeans found 4 clusters and Mean Shift didn't find any clusters. The clusters can be used as new features that may help to boost the prediction to catch bad loans.

# Summary

- SARIMA predicted 8 weeks and forecasted 12 weeks of rejected loan applications. The model had an RMSE of 10,025 which is not bad as there are an average of 200k loans per week. To test the model, I would like to get the July loan data to see if the forecast was accurate.
- LSTM predicted the 8 weeks of loans too, and had a RMSE of 13,033.
- SARIMA model was 1.54% better at predicting bad loans than LSTM.
- Lastly, it would be interesting to use my own information to see what category I would fall into.

# In Production

In production the model would give more time to LendingClub to look at the loans that were predicted as bad but were actually not bad which would increase revenue.

LendingClub would know the number of weekly loans predicted to be paid off therefore making sure they have resources to handle the collection processes.

They would find potentially new attributes of customers that are not likely to pay off their loans, the 4 new Kmeans clusters. customers who have a high credit limit that is over 30% of their income.

# Questions & Suggestion?



[https://  
www.linkedin.com/in/  
charla-gaddy/](https://www.linkedin.com/in/charla-gaddy/)