Predict User Propensity to Default on Loans
Binary Classification

I used the provided dataset to predict the user propensity to default on loans. The challenges were dealing with an imbalanced dataset and missing data.

Dataset Review:

This data consists of 6604 rows and their activities and records as shown below.

- General data information

```
Data columns (total 18 columns):
client_id                       6604 non-null int64
pit                             6604 non-null object
period                          6604 non-null object
recent_successful_repayments    6562 non-null float64
future_bad_probability          6604 non-null float64
available_credit                6484 non-null float64
balance                         6502 non-null float64
sum_failed_repayments           6368 non-null float64
max_failed_repayments           6368 non-null float64
credit_score                    5809 non-null float64
external score                  6604 non-null float64
credit_inquiries_count          5782 non-null float64
credit_open_balance             5753 non-null float64
years_on_file                   5406 non-null object
max_successful_repayments       3480 non-null float64
missing_report                  6604 non-null int64
client_industry_unknown         6604 non-null int64
tag_in_six_months               6604 non-null object
dtypes: float64(11), int64(3), object(4)
```

- Target/Label:
    - Tag_in_six_months: bad (=0) and good (=1)

- Columns with missing data:
    - recent_successful_repayments
    - available_credit
    - balance
    - sum_failed_repayments
    - max_failed_repayments
    - credit_score
    - credit_inquiries_count
    - credit_open_balance
    - years_on_file (converted “nan” to 0)
    - max_successful_repayments

Data Exploration:

- Addressed missing data

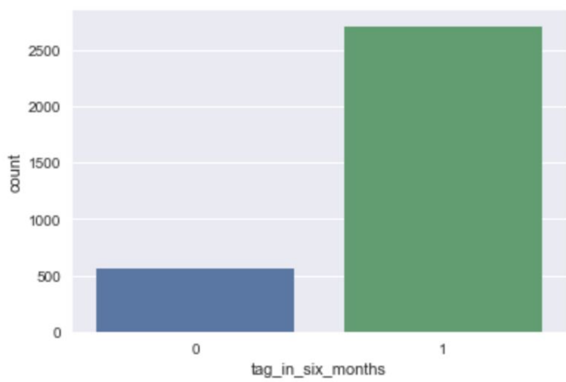
Rather than simply using the column mean, median, etc. for the missing values, I looped over columns with missing values and used a random forest regressor (or classifier, depending on the data type).

- Imbalanced data

The distribution of the target variable shows that two classes (good = 1 and bad = 0) are imbalanced.

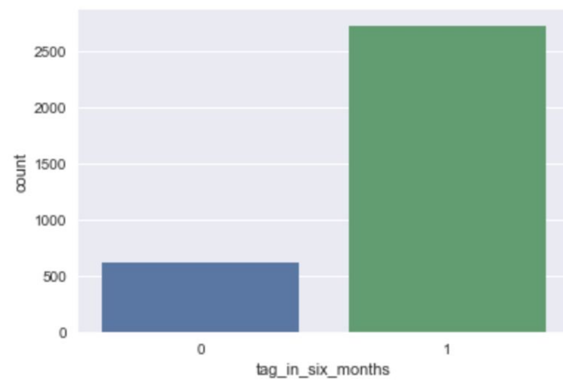
```
sns.countplot(df_120['tag_in_six_months'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a229f72b0>
```




```
sns.countplot(df_180['tag_in_six_months'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26084c88>
```



For very imbalanced data sets, it is often the case that machine learning algorithms will have a tendency to always predict the more dominant class when presented with new, unseen test data. To balance the classes, rather than simply oversampling the the minority class or undersampling the dominant class, I used the Synthetic Minority Oversampling Technique (SMOTE).

- Dropped highly correlated/redundant data to address multicollinearity

```
cor = df.corr()
cor.loc[:,:] = np.tril(cor, k=-1) # below main lower triangle of an array
cor = cor.stack()
cor[(cor > 0.55) | (cor < -0.55)]
```

```
max_failed_repayments  sum_failed_repayments     0.781213
dtype: float64
```

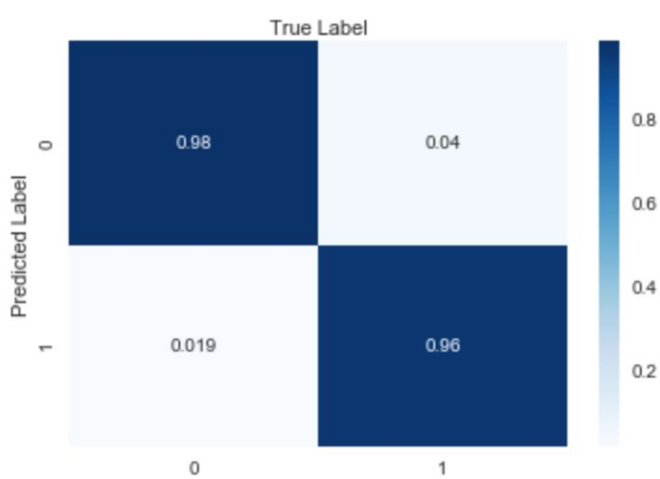
- Converted strings to numerical representations where possible
- Dropped superfluous attributes

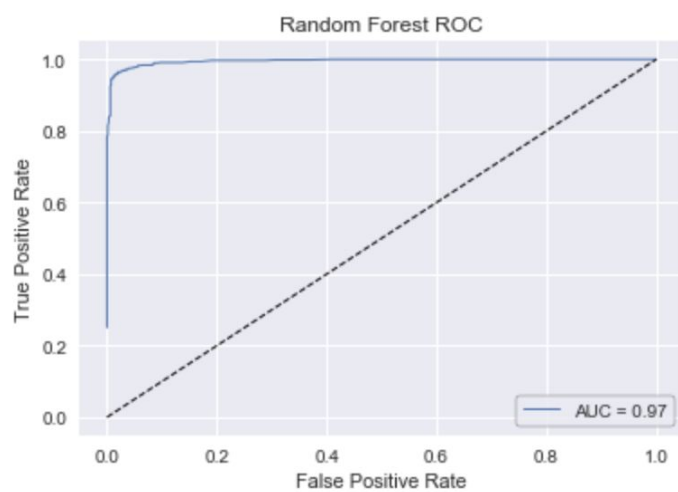
Model:

I used the python Sickit-Learn package for my prediction. These are the general steps I took to build the model.

- Divided the dataset two to datasets
    - 120 period
    - 180 period
- Filled NaNs for both datasets
- Balanced two datasets
- Created initial model using the training sets for both datasets
    - Random Forest Classifier
- Evaluated the model obtained using the test set
    - 120 dataset

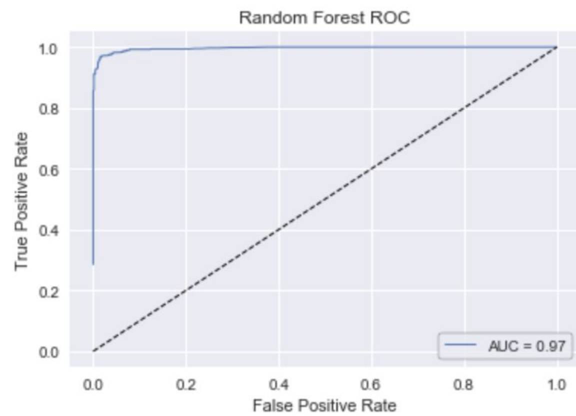
	precision	recall	f1-score	support
0	0.96	0.98	0.97	676
1	0.98	0.96	0.97	678
avg / total	0.97	0.97	0.97	1354



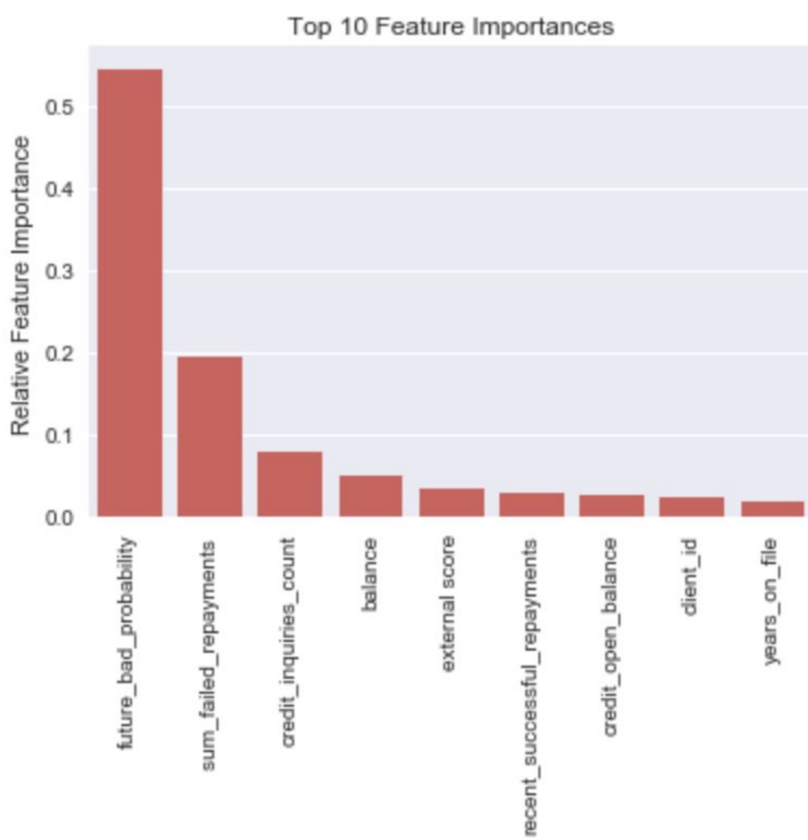


- 180 dataset

	precision	recall	f1-score	support
0	0.97	0.98	0.97	685
1	0.98	0.97	0.97	679
avg / total	0.97	0.97	0.97	1364

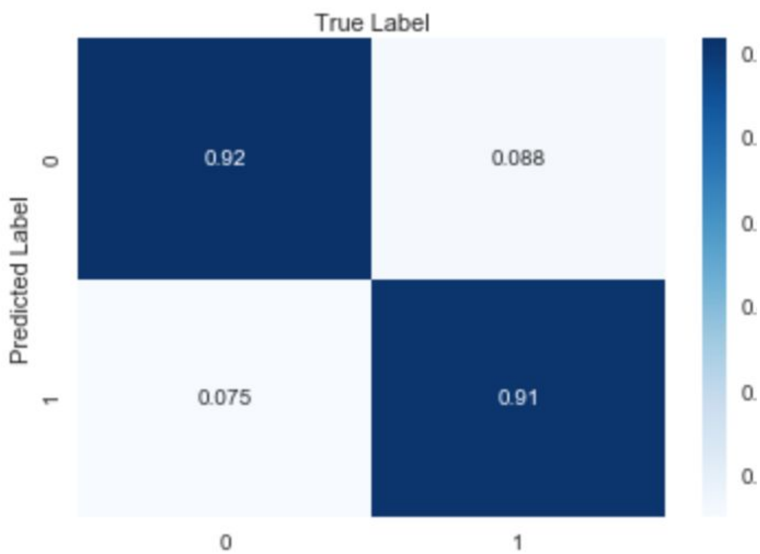


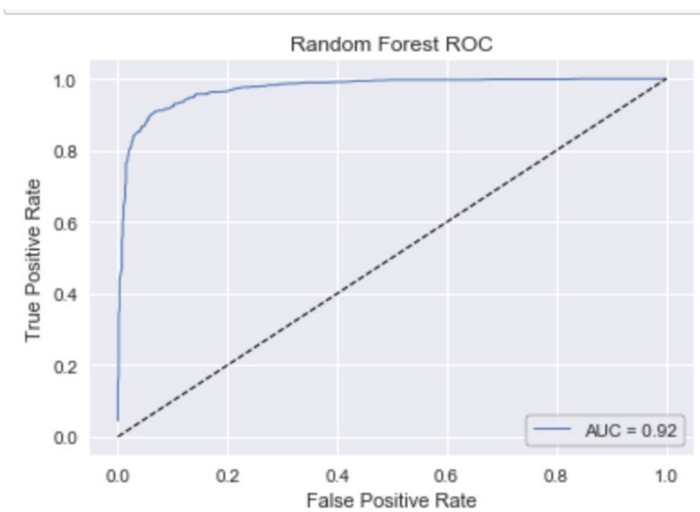
After I created my model I made a graph of most important features



Then decided to evaluate my model when I drop the future_bad_probability to make sure my model is not relying too much on the provided probability. For 120 period datasets the results are as below.

	precision	recall	f1-score	support
0	0.91	0.92	0.92	676
1	0.92	0.91	0.92	678
avg / total	0.92	0.92	0.92	1354





More Work:

- Perform data engineering and creating new features
- Request more information such as information about Demographic factors