Course Project on Supervised Machine Learning: Classification

Main objective of the analysis

The objective of this analysis is to develop classification model for prediction of loan payoffs depending on the loan amount, payoff schedule, and some information about applicants (their age, education, gender).

Brief description of the data set and a summary of its attributes

The data set used in this project includes information about past loans. If contains information about 400 customers whose loan are already paid off or defaulted. It includes following fields

Field	Description
Loan_status	Whether a loan is paid off or in collection
Principal	Basic principal loan amount
Terms	Origination terms which can be weekly (7 days), biweekly (15 days), and monthly (30 days) payoff schedule
Effective_date	When the loan got originated and took effects
Due_date	Since it's one-time payoff schedule, each loan has one single due date
Age	Age of applicant
Education	Education of applicant
Gender	The gender of applicant

For this project, I used two data sets which are used in the project for "Machine Learning with Python" course – loan_train.csv and loan_test.csv. The first data set contains information about 346 customers and the second one contains information about 54 customers. I joined both data sets together and then did my own split into training and testing data sets.

(Although I used data sets from that project, I did a completely new analysis in this project.)

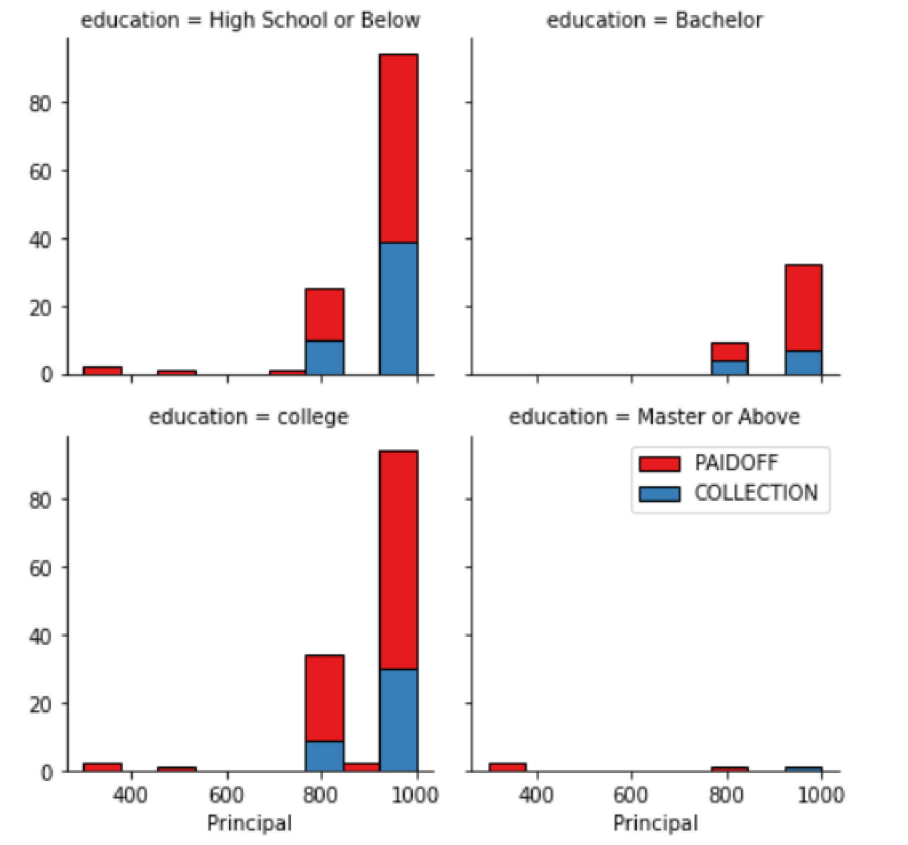
Brief summary of data exploration and actions taken for data cleaning and feature engineering

Data exploration includes data collection, data cleaning, analysis, and feature engineering.

Data Collection. I downloaded the data sets from Internet, examined them and joined them together into one data set.

Analysis. This included plotting some diagrams.

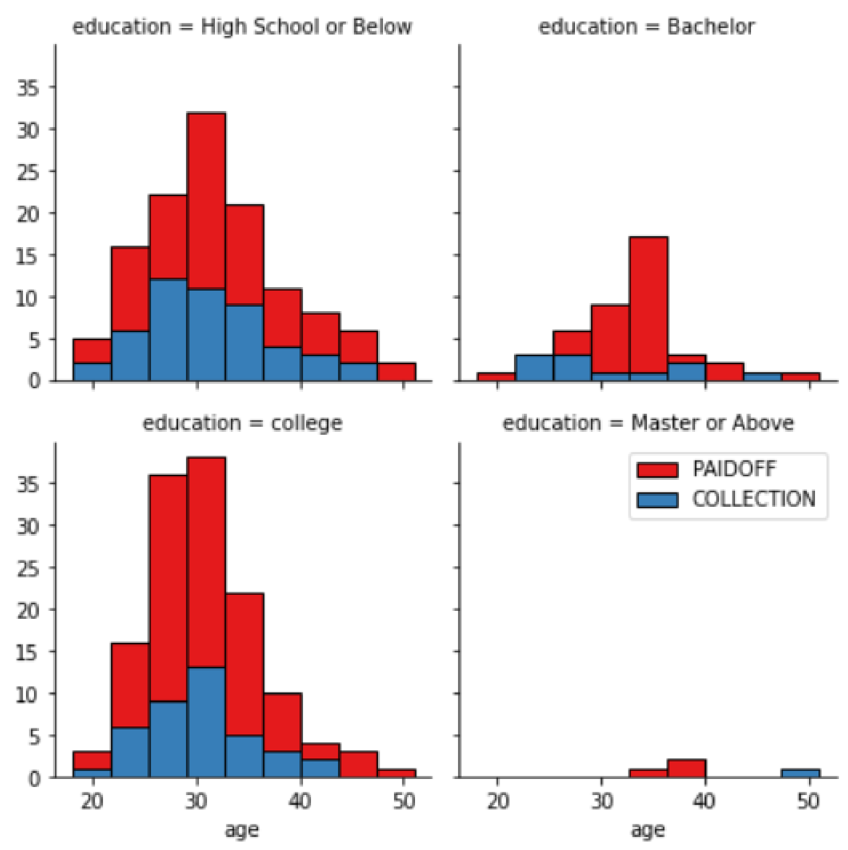
1. Education and the amount of loan (Principal) vs paid off:



These diagrams show that people who have lower education are much more likely to take loans than people with higher education. However loan status (paid off or in collection) does not so much depends on the level of education:

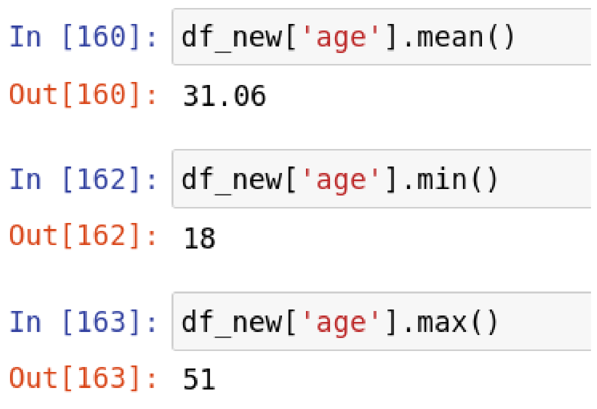
```
In [152]: df_new.groupby(['education'])['loan_status'].value_counts(normalize=True)

Out[152]: education             loan_status
          Bachelor              PAIDOFF        0.788462
                                COLLECTION     0.211538
          High School or Below  PAIDOFF        0.715116
                                COLLECTION     0.284884
          Master or Above       PAIDOFF        0.750000
                                COLLECTION     0.250000
          college               PAIDOFF        0.773256
                                COLLECTION     0.226744
          Name: loan_status, dtype: float64
```

2. Education and age vs paid off:



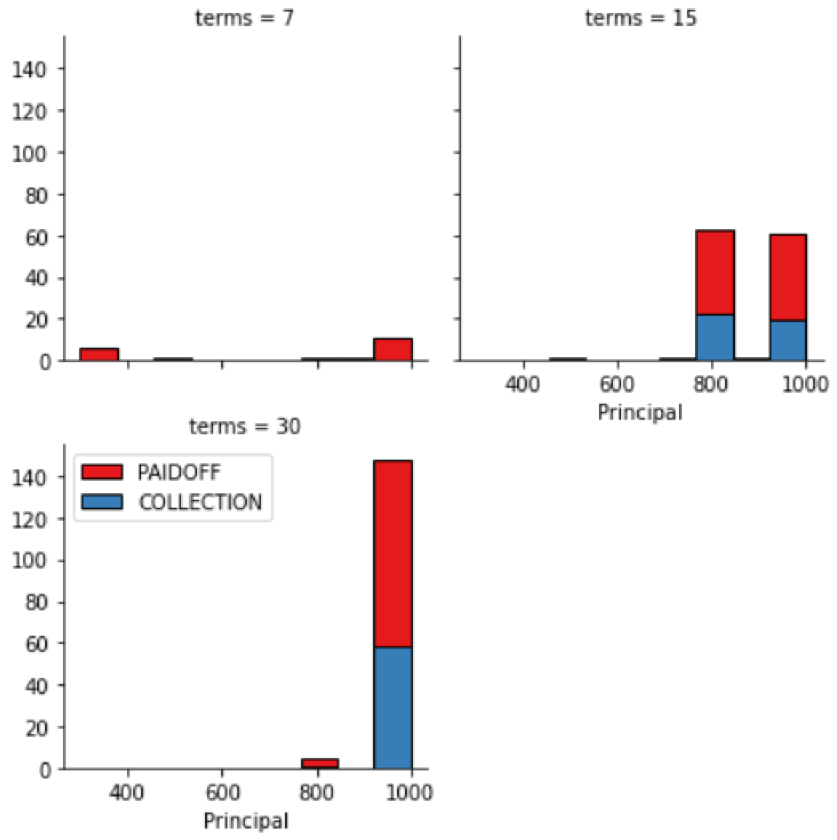
These diagrams show that most people who take loans are around 30 years old.

The mean age of customers is 31.06, minimum age is 18, maximum age is 51:



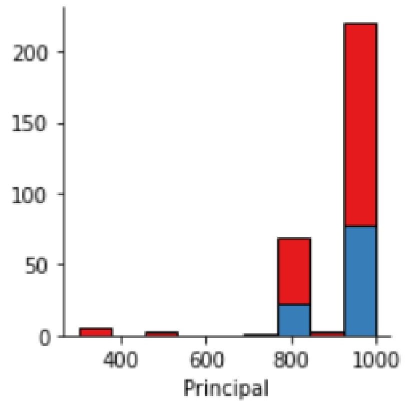
```
In [160]: df_new['age'].mean()

Out[160]: 31.06

In [162]: df_new['age'].min()

Out[162]: 18

In [163]: df_new['age'].max()

Out[163]: 51
```

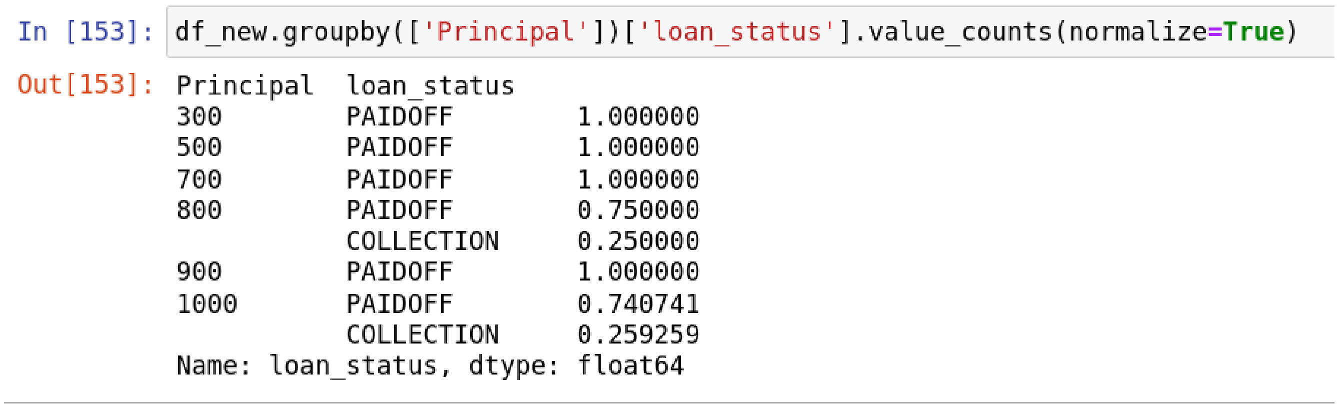
3. Terms (origination terms which can be weekly (7 days), biweekly (15 days), and monthly (30 days) payoff schedule) and loan amount (Principal) vs loan status:



These diagrams show that most people prefer monthly payoff schedule, but people who are on weekly payoff schedule are much more likely to pay off in time.

4. Loan amount vs. loan status:

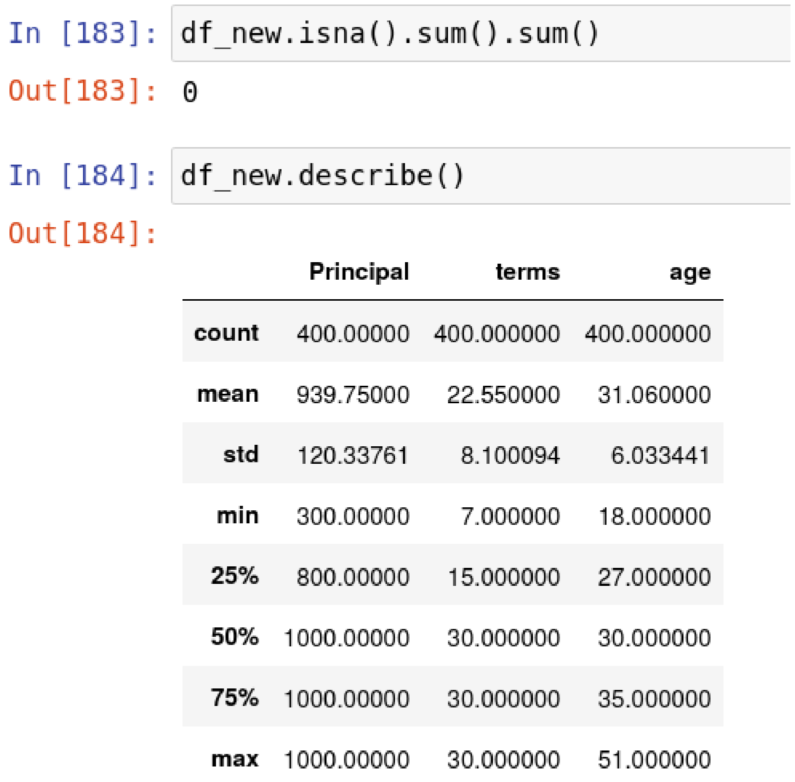




```
In [153]: df_new.groupby(['Principal'])['loan_status'].value_counts(normalize=True)

Out[153]: Principal  loan_status
          300        PAIDOFF         1.000000
          500        PAIDOFF         1.000000
          700        PAIDOFF         1.000000
          800        PAIDOFF         0.750000
                     COLLECTION      0.250000
          900        PAIDOFF         1.000000
          1000       PAIDOFF         0.740741
                     COLLECTION      0.259259
          Name: loan_status, dtype: float64
```

People who take smaller amounts of loans are more likely to pay off in time.

Data cleaning. This data set does not need data cleaning because it does not contain missing values and outliers:



```
In [183]: df_new.isna().sum().sum()

Out[183]: 0

In [184]: df_new.describe()

Out[184]:
```

	Principal	terms	age
count	400.00000	400.000000	400.000000
mean	939.75000	22.550000	31.060000
std	120.33761	8.100094	6.033441
min	300.00000	7.000000	18.000000
25%	800.00000	15.000000	27.000000
50%	1000.00000	30.000000	30.000000
75%	1000.00000	30.000000	35.000000
max	1000.00000	30.000000	51.000000

Feature engineering. It included the following steps:

- Converting dates (effective dates and due dates) into datetime objects.

- Converting gender from categorical to numerical values.
- One-hot encoding of education.
- Data normalization.
- Split to train and test data sets.

Summary of training classification models

In this project, I built and trained the following classification models:

- Simple logistic regression
- Logistic regression with L1 regularization
- Logistic regression with L2 regularization
- K-nearest neighbors with k = 3
- K-nearest neighbors with k = 5
- Support vector machine with linear kernel
- Support vector machine with RBF (radial basis function) kernel
- Decision tree
- Random forest
- Voting classifier with logistic regression (with L2 regularization) and random forest
- Voting classifier with KNN (k = 3) and random forest

For all the models I used the same training and test splits.

For evaluation of all the models I used F1-score and Jaccard similarity coefficient score.

1. Logistic regression models

1.1. Simple logistic regresssion model

In [161]:
```
# Building and training Logistic Regression model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score

lr = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
yhat_lr = lr.predict(X_test)
print("F1 score:", round(f1_score(y_test, yhat_lr, average='weighted'), 4))
print("Jaccard score:", round(jaccard_score(y_test, yhat_lr, pos_label='PAIDOFF'), 4))
```

```
F1 score: 0.6914
Jaccard score: 0.7857
```

1.2. Logistic regression with L1 regularization

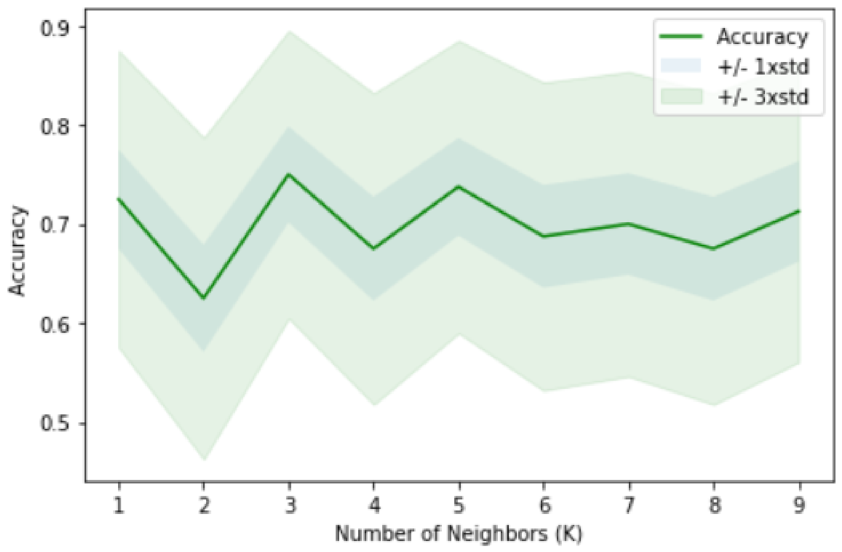
In [162]:
```
# L1 regularized logistic regression
from sklearn.linear_model import LogisticRegressionCV
lr_l1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear').fit(X_train, y_train)
yhat_lr_l1 = lr_l1.predict(X_test)
print("F1 score:", round(f1_score(y_test, yhat_lr_l1, average='weighted'), 4))
print("Jaccard score:", round(jaccard_score(y_test, yhat_lr_l1, pos_label='PAIDOFF'), 4))
```

```
F1 score: 0.7143
Jaccard score: 0.75
```

1.3. Logistic regression with L2 regularization

In [163]:
```
# L2 regularized logistic regression
lr_l2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2', solver='liblinear').fit(X_train, y_train)
yhat_lr_l2 = lr_l2.predict(X_test)
print("F1 score:", round(f1_score(y_test, yhat_lr_l2, average='weighted'), 4))
print("Jaccard score:", round(jaccard_score(y_test, yhat_lr_l2, pos_label='PAIDOFF'), 4))
```

```
F1 score: 0.7048
Jaccard score: 0.7353
```



For this data, the best accuracy will give KNN with k=3. KNN with k=5 will give a little worse accuracy. So, I chose KNN with k=3 and k=5.

2. K-nearest neighbors

2.1. K=3 ¶

```
In [148]: #K-nearest neighbors
          from sklearn.neighbors import KNeighborsClassifier
          knn3 = KNeighborsClassifier(n_neighbors = 3).fit(X_train,y_train)
          y_pred = knn3.predict(X_test)
          print('F1 Score: ', round(f1_score(y_test, y_pred, average='weighted'), 4))
          print('Jaccard: ', round(jaccard_score(y_test, y_pred, pos_label='PAIDOFF'), 4))
```

```
F1 Score:  0.7143
Jaccard:  0.6923
```

2.2 K=5

```
In [149]: knn5 = KNeighborsClassifier(n_neighbors = 5).fit(X_train,y_train)
          y_pred = knn5.predict(X_test)
          print('F1 Score: ', round(f1_score(y_test, y_pred, average='weighted'), 4))
          print('Jaccard: ', round(jaccard_score(y_test, y_pred, pos_label='PAIDOFF'), 4))
```

```
F1 Score:  0.754
Jaccard:  0.7344
```

3. Support vector machine

3.1. SVM with linear kernel

```
In [150]: # Building SVM model and training it
          from sklearn.svm import LinearSVC
          LSVC = LinearSVC().fit(X_train, y_train)
          yhat_lsvc = LSVC.predict(X_test)
          print("Jaccard score:", round(jaccard_score(y_test, yhat_lsvc, pos_label='PAIDOFF'), 4))
          print("F1 score", round(f1_score(y_test, yhat_lsvc, average='weighted'), 4))
```

```
Jaccard score: 0.7206
F1 score 0.6954
```

3.2. SVM with non-linear (RBF) kernel

```
In [152]: from sklearn.svm import SVC
          RBF = SVC(kernel='rbf').fit(X_train, y_train)
          yhat_rbf = RBF.predict(X_test)
          print("Jaccard score:", round(jaccard_score(y_test, yhat_rbf, pos_label='PAIDOFF'), 4))
          print("F1 score", round(f1_score(y_test, yhat_rbf, average='weighted'), 4))
```

```
Jaccard score: 0.7273
F1 score 0.7276
```

4. Decision tree ¶

```
In [153]: # Building and training Decision Tree model
          from sklearn.tree import DecisionTreeClassifier
          X_train, y_train = X, y
          dt = DecisionTreeClassifier(random_state=42, criterion="entropy", max_depth = 4).fit(X_train, y_train)
          yhat_dt = dt.predict(X_test)
          print("Jaccard score:", round(jaccard_score(y_test, yhat_dt, pos_label='PAIDOFF'), 4))
          print("F1 score:", round(f1_score(y_test, yhat_dt, average='weighted'), 4))
```

```
Jaccard score: 0.7681
F1 score: 0.7065
```

	oob
n_trees	
15.0	0.3125
20.0	0.3050
30.0	0.3075
40.0	0.2975
50.0	0.3025
100.0	0.2950
150.0	0.2900
200.0	0.2975
300.0	0.3000
400.0	0.2975

For Random Forest Classifier, the minimal out-of-bag error (oob) is when n_trees = 100.

5. Random forest ¶

```
In [157]: from sklearn.ensemble import RandomForestClassifier
          RF = RandomForestClassifier(n_estimators = 100, random_state=42).fit(X_train, y_train)
          y_pred = RF.predict(X_test)
          print("Jaccard score:", round(jaccard_score(y_test, y_pred, pos_label='PAIDOFF'), 4))
          print("F1 score:", round(f1_score(y_test, y_pred, average='weighted'), 4))
```

```
Jaccard score: 0.8644
F1 score: 0.8857
```

6. Voting classifiers

6.1. Voting classifier with logistic regression with L2 and random forest ¶

In [159]:
```
from sklearn.ensemble import VotingClassifier

# The combined model--logistic regression with L2 and random forest
estimators1 = [('lr_l2', lr_l2), ('RF', RF)]
VC1 = VotingClassifier(estimators1, voting='soft')
VC1 = VC1.fit(X_train, y_train)
y_pred = VC1.predict(X_test)
print("Jaccard score:", round(jaccard_score(y_test, y_pred, pos_label='PAIDOFF'), 4))
print("F1 score", round(f1_score(y_test, y_pred, average='weighted'), 4))
```

```
Jaccard score: 0.8644
F1 score 0.8857
```

6.2. Voting classifier with KNN (k=3) and random forest

In [160]:
```
# The combined model--knn and random forest
estimators2 = [('knn3', knn3), ('RF', RF)]
VC2 = VotingClassifier(estimators2, voting='soft')
VC2 = VC2.fit(X_train, y_train)
y_pred = VC2.predict(X_test)
print("Jaccard score:", round(jaccard_score(y_test, y_pred, pos_label='PAIDOFF'), 4))
print("F1 score", round(f1_score(y_test, y_pred, average='weighted'), 4))
```

```
Jaccard score: 0.8667
F1 score 0.8827
```

The summary of evaluation of all models:

Model	F1-score	Jaccard
logistic regression (LR)	0.7401	0.7632
LR with L1 regularization	0.6768	0.7750
LR with L2 regularization	0.7775	0.7838
KNN (k=3)	0.7143	0.6923
KNN (k=5)	0.754	0.7344
SVM with linear kernel	0.6954	0.7206
SVM with rbf kernel	0.7276	0.7273
decision tree	0.7065	0.7681
random forest (RF)	0.8857	0.8644
voting classifier (LR_L2, RF)	0.8857	0.8644
voting classifier (knn3, RF)	0.8827	0.8667

Recommended final model

Among all the models used in this project, the random forest model and the voting classifier with linear regression (with L2 regularization) and random forest give the highest F1-score. The voting classifier with KNN (n=3) and random forest has higher Jaccard score, but lower F1-score.

These three models have much higher scores than all the other models.

Since these three models have very similar score, probably, it would be better to use the random forest model since it is more simple than the voting classifiers and its scores are practically the same as of those more complicated models.

Summary Key Findings and Insights

The goal of this project was to develop a classification model that can be used to predict what kind of customers are more likely to pay off loans.

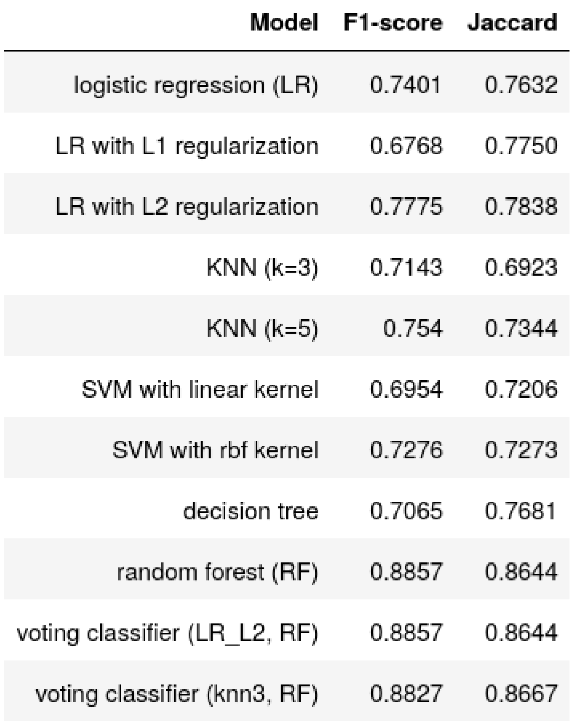
Exploratory descriptive analysis of the data set used in this project, showed that there is correlation between loan status (paidoff or in collection) and all the features. Thus, the higher the amount of loan is, the less customers are likely to pay off in time. Those customers who are required to pay off weekly, are more likely to pay off in time than those who are required to pay off monthly. If the due date is Friday or weekend, customers are much more likely to miss the due date of payment than if the due date is a weekday. People with higher education are much less likely to take loans than people with lower education, but people with lower education are less likely to pay loans. In general, younger people are more likely to miss the due date than older people. Women are more likely to pay loans than men.

Since all these features affect the loan status, classification models need to be trained on all of those features.

In this project, I built 11 classification models: three logistic regression models, two K-nearest neighbors models, two SVM models, decision tree, random forest, and two voting classifier models.

I used two metrics for evaluating all the models – F1-score and Jaccard score. As could be expected, more simple models have lower scores than more complicated models. Random forest and voting classifiers have much better scores than other models.

The table of F1-scores and Jaccard scores for all the models is shown on the next page.



Model	F1-score	Jaccard
logistic regression (LR)	0.7401	0.7632
LR with L1 regularization	0.6768	0.7750
LR with L2 regularization	0.7775	0.7838
KNN (k=3)	0.7143	0.6923
KNN (k=5)	0.754	0.7344
SVM with linear kernel	0.6954	0.7206
SVM with rbf kernel	0.7276	0.7273
decision tree	0.7065	0.7681
random forest (RF)	0.8857	0.8644
voting classifier (LR_L2, RF)	0.8857	0.8644
voting classifier (knn3, RF)	0.8827	0.8667

Suggestions for next steps in analyzing this data

Although random forest and voting classifiers have much higher scores than other models probably it is still possible to built a more accurate classification model by tuning parameters or using boosting or stacking models.

Also, the data set used in this project is not very large. It contains only 400 examples. Model accuracy can also be improved by using more examples for training.

Another possible way to increase model performance is to use larger data sets with more features and more examples. The features that are in the data set used in this project are not the only features that affect people’s paying loans. If model is trained on more features, its performance will also increased.