Home Loan Repayment

Group 4: Yuexuan Fu, Anika Kingsley, Henry Raymond, Xinye Xu, Yuxin Zhang

Presentation Contents

- Project Description
  - Data Description
  - Business insight
  - Project Scope
- Analysis
  - 4 Classification Models
- Summary of Findings
  - Conclusion
  - Recommendations

Project Description

Data

Binary target variable:

- 1 - client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan in our sample,
- 0 - all other cases

Features: client specific features (credit, employment, family size, etc.)

Data Size:

```
train.shape
(307511, 122)
```

Business Insight

Default Predictions

- False negative: costly
- False positive: not as costly

Loaning to those with poor credit history

- Analyze alternative data
- Minimize false negatives with recall score

Data Cleaning

- Dropped columns with N/A values
- Filled N/A with means/0
- Dropped features
    - Irrelevant to our business question
    - Mismatching data
- Turned categorical variables into binary
- Merged data (Add Previous Loans Count

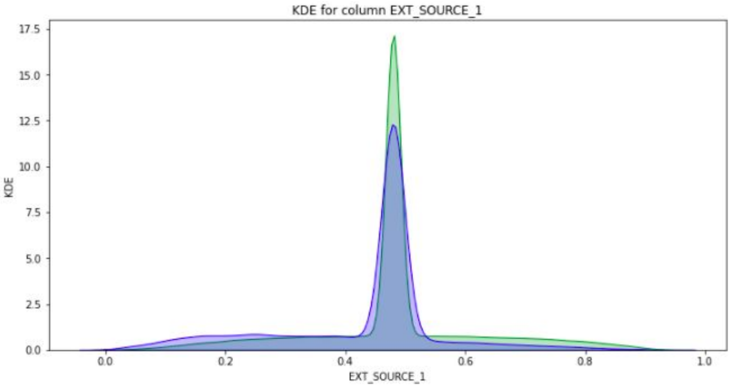
  from Bureau.csv on SK-ID-CURR)

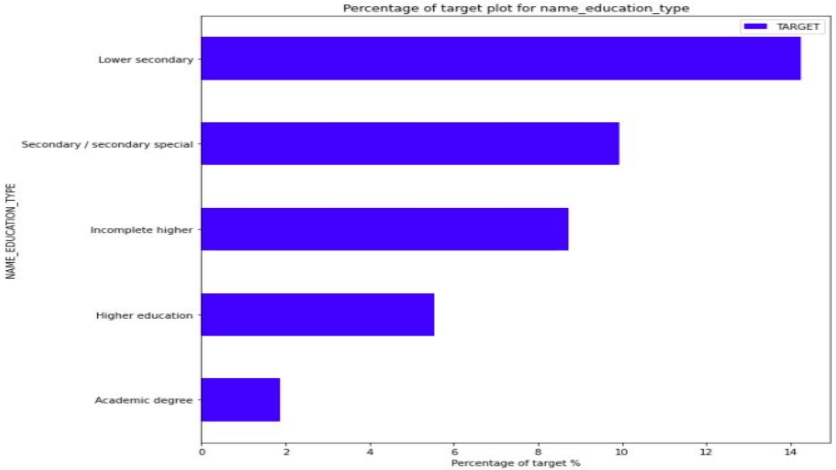
After:

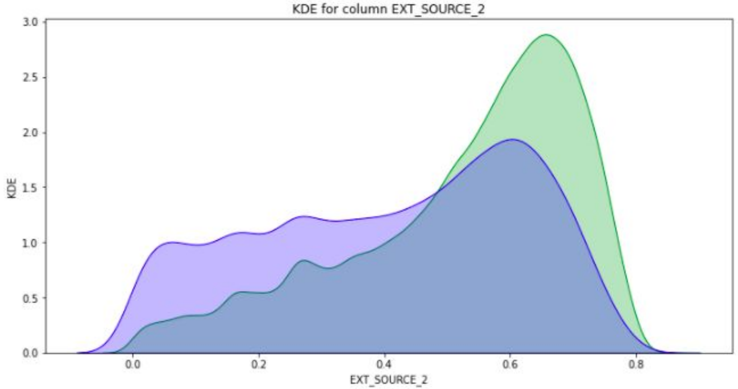
	dtypes	column count
0	object	13
1	float64	10
2	int64	9

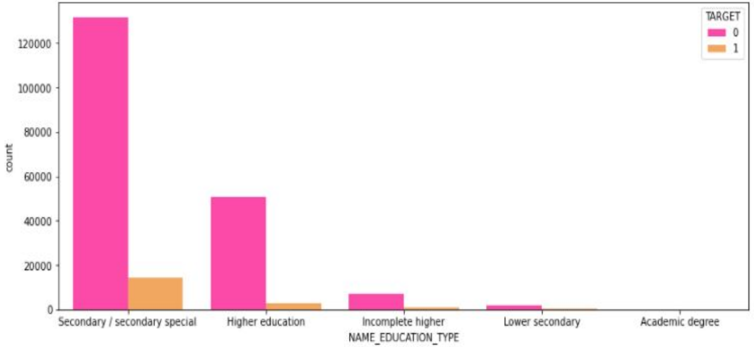
	Number of Missing Values	Percentage of Entries Missing
COMMONAREA_MEDI	214865	69.87
COMMONAREA_AVG	214865	69.87
COMMONAREA_MODE	214865	69.87
NONLIVINGAPARTMENTS_MODE	213514	69.43
NONLIVINGAPARTMENTS_AVG	213514	69.43
NONLIVINGAPARTMENTS_MEDI	213514	69.43
FONDKAPREMONT_MODE	210295	68.39
LIVINGAPARTMENTS_MODE	210199	68.35
LIVINGAPARTMENTS_AVG	210199	68.35
LIVINGAPARTMENTS_MEDI	210199	68.35
FLOORSMIN_AVG	208642	67.85
FLOORSMIN_MODE	208642	67.85
FLOORSMIN_MEDI	208642	67.85

Exploratory Data Analysis - Important Features

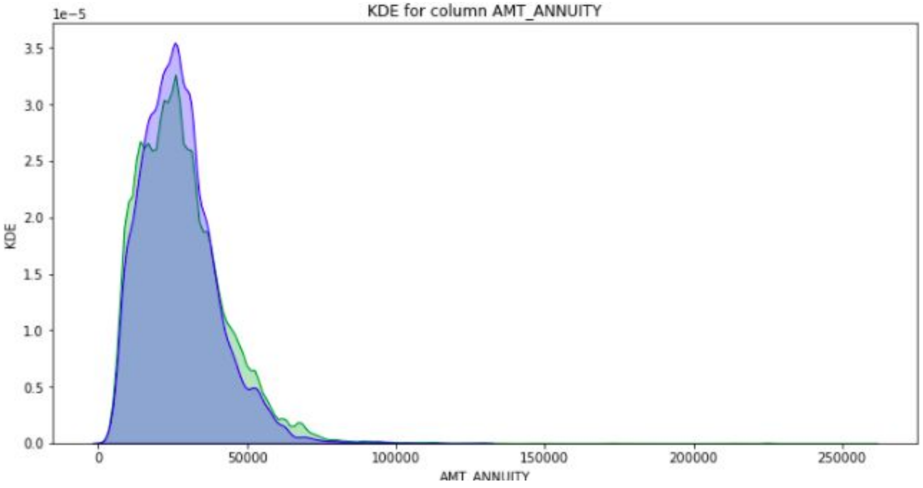




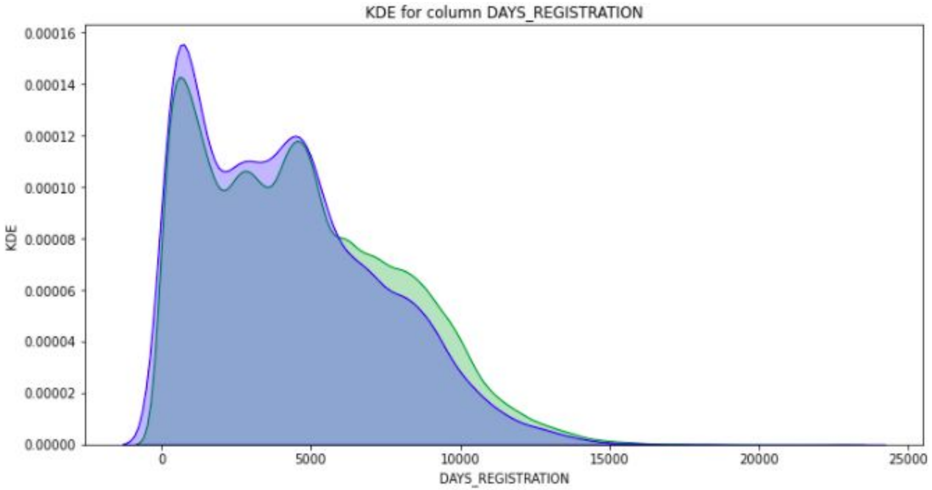




Exploratory Data Analysis - Less Important Features

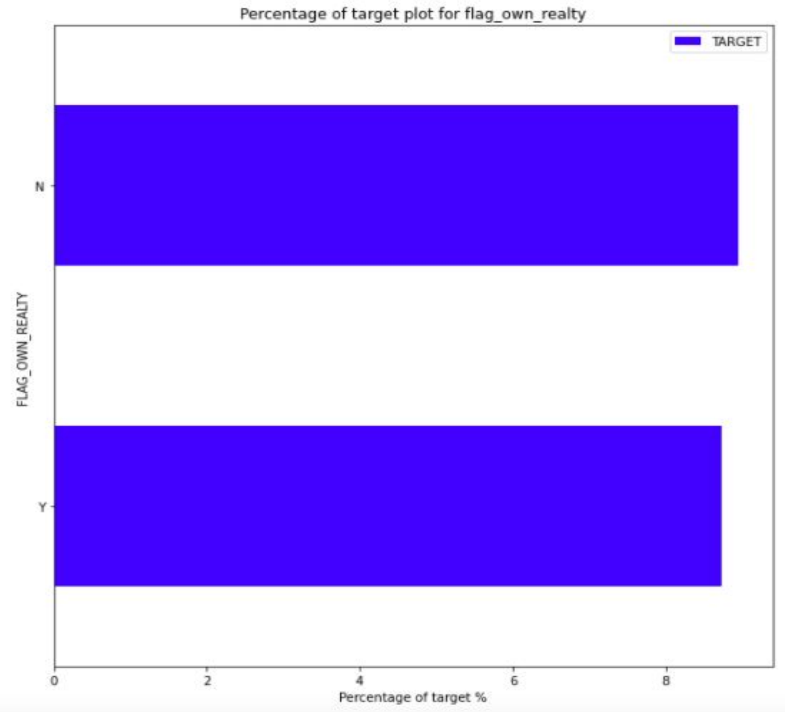


Loan annuity



How many days before the application did client change his registration

Exploratory Data Analysis - Features that seems with no impact



Flag if client owns a house or flat

Business Question:

What attributes of a client are associated with loan repayment?

Project Scope

Determine significant predictors of whether a client will repay a loan by using the following models:

- Logistic regression
- Linear SVC
- Decision Tree
- Random Forest

Analysis

```python
safe= train[train.TARGET == 0]
bad= train[train.TARGET == 1]
percentage = len(bad)/float(len(safe))
percentage
risky_loans = bad
safe_loans = safe.sample(frac = percentage, random_state = 1)
print( "Number of safe loans  : " , len(safe_loans))
print( "Number of risky loans : " , len(risky_loans))
# Append the risky_loans with the downsampled version of safe_loans
train = risky_loans.append(safe_loans)
```

```
Number of safe loans  :  3689
Number of risky loans :  3689
```

Logistic Regression

```
col_cat = ['NAME_CONTRACT_TYPE', 'CODE_GENDER','FLAG_OWN_CAR', 'FLAG_OWN_REALTY',
           'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
           'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE', 'ORGANIZATION_TYPE']
col_num = ['SK_ID_CURR', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY',
           'AMT_GOODS_PRICE',  'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH','DAYS_EMPLOYED',
           'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH','CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT',
           'REGION_RATING_CLIENT_W_CITY', 'LIVE_REGION_NOT_WORK_REGION','EXT_SOURCE_1',
           'EXT_SOURCE_2', 'EXT_SOURCE_3','AMT_REQ_CREDIT_BUREAU_QRT', 'previous_loan_counts']

X_train_cat = X_train[col_cat]
X_train_num = X_train[col_num]
X_test_cat = X_test[col_cat]
X_test_num = X_test[col_num]
```

Logistic Regression

```
pipe_cat = make_pipeline(SimpleImputer(strategy='constant'), OneHotEncoder(handle_unknown='ignore'))
pipe_num = make_pipeline(SimpleImputer(),MinMaxScaler())
preprocessor = make_column_transformer((pipe_cat, col_cat), (pipe_num, col_num))
```

```
make_pipeline(preprocessor, LogisticRegression(solver='lbfgs', multi_class='auto', random_state=42, max_iter=1000))
```

```
pipe.fit(X_train, y_train)
accuracy = pipe.score(X_test, y_test)
print('Accuracy score of the {} with MinMaxScaler is {:.3f}'.format(pipe.__class__.__name__, accuracy))
```

```
Accuracy score of the Pipeline with MinMaxScaler is 0.671
```

Cross Validation

```python
from sklearn.model_selection import cross_validate

pipe = make_pipeline(preprocessor, LogisticRegression(solver='lbfgs', multi_class='auto',
                                                      random_state=42, max_iter=1000))
scores = cross_validate(pipe, X, y, cv=3, return_train_score=True)
```

```python
df_scores = pd.DataFrame(scores)
df_scores
```

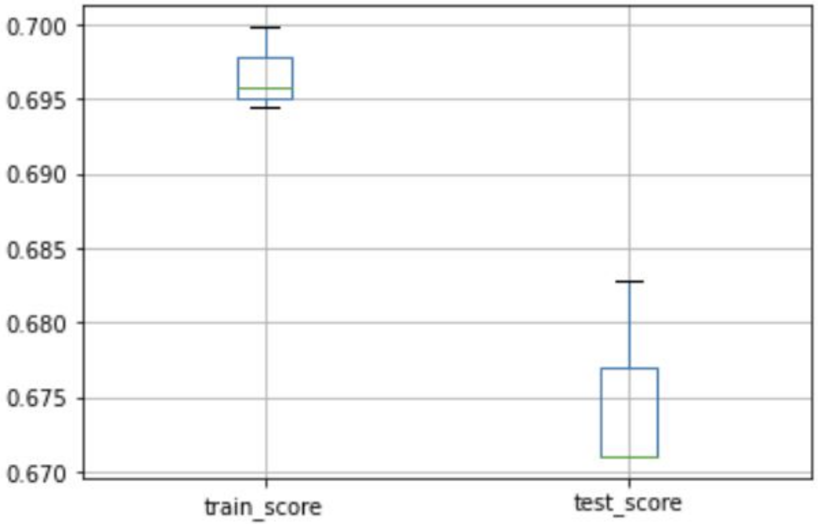
	fit_time	score_time	test_score	train_score
0	8.32	0.44	0.91	0.91
1	8.18	0.41	0.91	0.91
2	9.23	0.41	0.91	0.91

```
1 print("Mean times and scores:\n", df_scores.mean())
```

```
Mean times and scores:
 fit_time       0.39
score_time     0.03
test_score     0.67
train_score    0.70
dtype: float64
```

```
1 df_scores[['train_score', 'test_score']].boxplot()
```

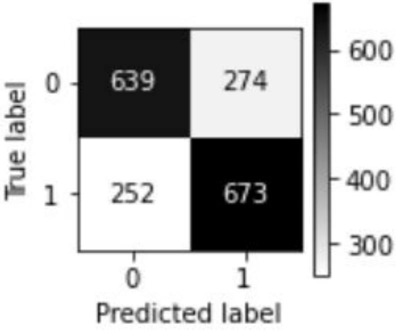
```
<AxesSubplot:>
```



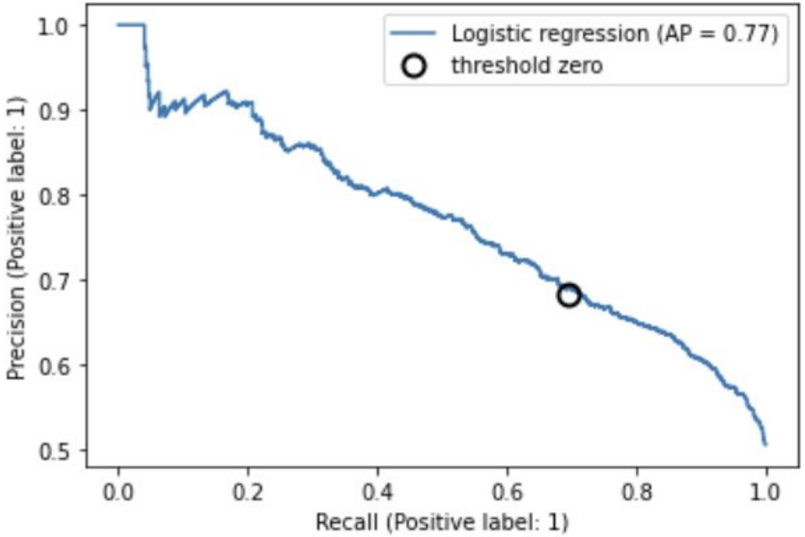
In testing set, the interquartile lies between 0.672 to 0.677, the mean is about 0.672.

```python
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,balanced_accuracy_score,roc_auc_sc

model_log_predicted = model_log.predict(X_test)

print('Accuracy of model_rf: {:.2f}'.format(accuracy_score(y_test, model_log_predicted)))
print('Precision of model_rf: {:.2f}'.format(precision_score(y_test, model_log_predicted)))
print('Recall of model_rf: {:.2f}'.format(recall_score(y_test, model_log_predicted)))
print('F1 of model_rf: {:.2f}'.format(f1_score(y_test, model_log_predicted)))
print('Balanced accuracy score of model_rf: {:.2f}'.format(balanced_accuracy_score(y_test, model_log_predicted)))
print('Roc auc score of model_rf: {:.2f}'.format(roc_auc_score(y_test, model_log_predicted)))
```

```
Accuracy of model_rf: 0.69
Precision of model_rf: 0.69
Recall of model_rf: 0.68
F1 of model_rf: 0.69
Balanced accuracy score of model_rf: 0.69
Roc auc score of model_rf: 0.69
```




```
<matplotlib.legend.Legend at 0x7fc23e30ddc0>
```



Linear SVC

```
from sklearn.svm import LinearSVC
clf = make_pipeline(preprocessor, LinearSVC(C=1.0, max_iter=500000))
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)
print('Accuracy score of the {} is {:.3f}'.format(clf.__class__.__name__, accuracy))
```

```
Accuracy score of the Pipeline is 0.675
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
pipe1 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=2, criterion='entropy'))
pipe2 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=6, criterion='entropy'))
pipe3 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=10, criterion='entropy'))
```

```
pipe_cat = make_pipeline(SimpleImputer(strategy='constant'), OneHotEncoder(handle_unknown='ignore'))
pipe_num = make_pipeline(SimpleImputer(),MinMaxScaler())
preprocessor = make_column_transformer((pipe_cat, col_cat), (pipe_num, col_num))
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
pipe1 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=2, criterion='entropy'))
pipe2 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=6, criterion='entropy'))
pipe3 = make_pipeline(preprocessor, DecisionTreeClassifier(max_depth=10, criterion='entropy'))


pipe1.fit(X_train, y_train)
accuracy = pipe1.score(X_test, y_test)
print('Accuracy score of the decision tree of depth 2 {} is {:.5f}'.format(pipe1.__class__.__name__, accuracy))

pipe2.fit(X_train, y_train)
accuracy = pipe2.score(X_test, y_test)
print('Accuracy score of the decision tree of depth 6 {} is {:.5f}'.format(pipe2.__class__.__name__, accuracy))

pipe3.fit(X_train, y_train)
accuracy = pipe3.score(X_test, y_test)
print('Accuracy score of the decision tree of depth 10 {} is {:.5f}'.format(pipe3.__class__.__name__, accuracy))
```

```
Accuracy score of the decision tree of depth 2 Pipeline is 0.62840
Accuracy score of the decision tree of depth 6 Pipeline is 0.65778
Accuracy score of the decision tree of depth 10 Pipeline is 0.62514
```

Random Forest

```
pipe_cat = make_pipeline(SimpleImputer(strategy='constant'), OneHotEncoder(handle_unknown='ignore'))
pipe_num = make_pipeline(SimpleImputer())
preprocessor = make_column_transformer((pipe_cat, col_cat), (pipe_num, col_num))
rf = RandomForestClassifier()
pipe_rf = make_pipeline(preprocessor, rf)
pipe_rf.get_params()
```

```
'randomforestclassifier__max_depth': None,
'randomforestclassifier__max_features': 'auto',
'randomforestclassifier__max_leaf_nodes': None,
'randomforestclassifier__max_samples': None,
'randomforestclassifier__min_impurity_decrease': 0.0,
'randomforestclassifier__min_impurity_split': None,
'randomforestclassifier__min_samples_leaf': 1,
'randomforestclassifier__min_samples_split': 2,
'randomforestclassifier__min_weight_fraction_leaf': 0.0,
'randomforestclassifier__n_estimators': 100,
'randomforestclassifier__n_jobs': None,
```

```python
from sklearn.ensemble import RandomForestClassifier
np.set_printoptions(precision=6, suppress=True)
param_grid = {'randomforestclassifier__max_features': [2,4,6,8,10],
              'randomforestclassifier__max_depth': [6,8,10,12,14]}
param_grid
```

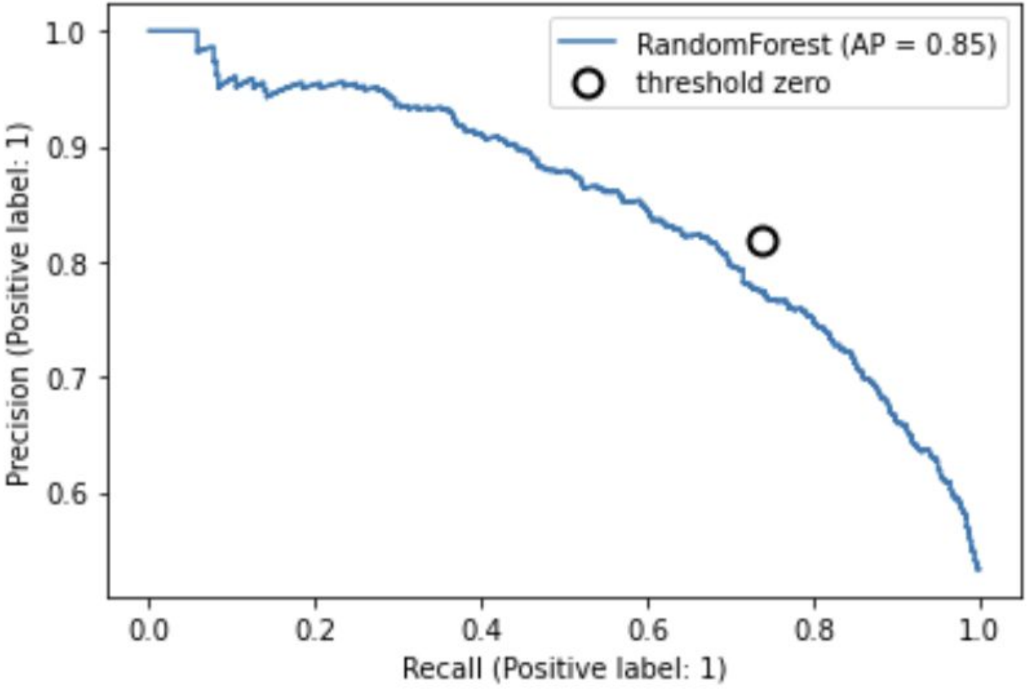
```
{'randomforestclassifier__max_features': [2, 4, 6, 8, 10],
 'randomforestclassifier__max_depth': [6, 8, 10, 12, 14]}
```

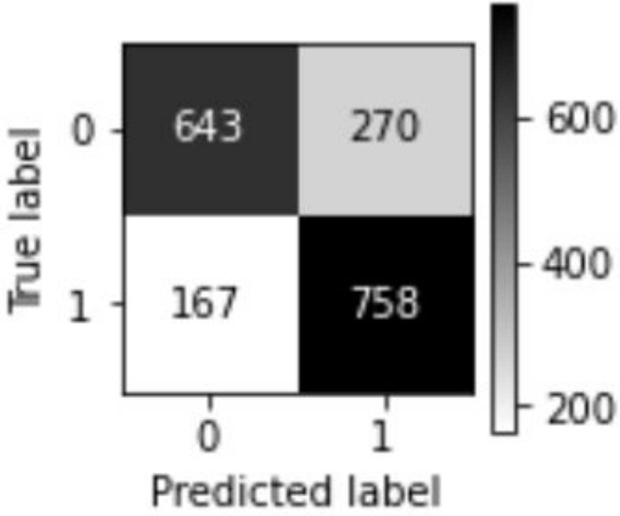
```python
from sklearn.model_selection import GridSearchCV

grid_rf = GridSearchCV(pipe_rf, param_grid=param_grid, cv=3, return_train_score=True)
grid_rf.fit(X_train, y_train)

print(f"best parameters: {grid_rf.best_params_}")

print(f"test-set score: {grid_rf.score(X_test, y_test):.3f}")
```

```
best parameters: {'randomforestclassifier__max_depth': 6, 'randomforestclassifier__max_features': 10}
test-set score: 0.690
```

```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,balanced_accuracy_score,roc_auc_sc

model_rf_predicted = model_rf.predict(X_test)

print('Accuracy of model_rf: {:.2f}'.format(accuracy_score(y_test, model_rf_predicted)))
print('Precision of model_rf: {:.2f}'.format(precision_score(y_test, model_rf_predicted)))
print('Recall of model_rf: {:.2f}'.format(recall_score(y_test, model_rf_predicted)))
print('F1 of model_rf: {:.2f}'.format(f1_score(y_test, model_rf_predicted)))
print('Balanced accuracy score of model_rf: {:.2f}'.format(balanced_accuracy_score(y_test, model_rf_predicted)))
print('Roc auc score of model_rf: {:.2f}'.format(roc_auc_score(y_test, model_rf_predicted)))
```

```
Accuracy of model_rf: 0.76
Precision of model_rf: 0.74
Recall of model_rf: 0.82
F1 of model_rf: 0.78
Balanced accuracy score of model_rf: 0.76
Roc auc score of model_rf: 0.76
```

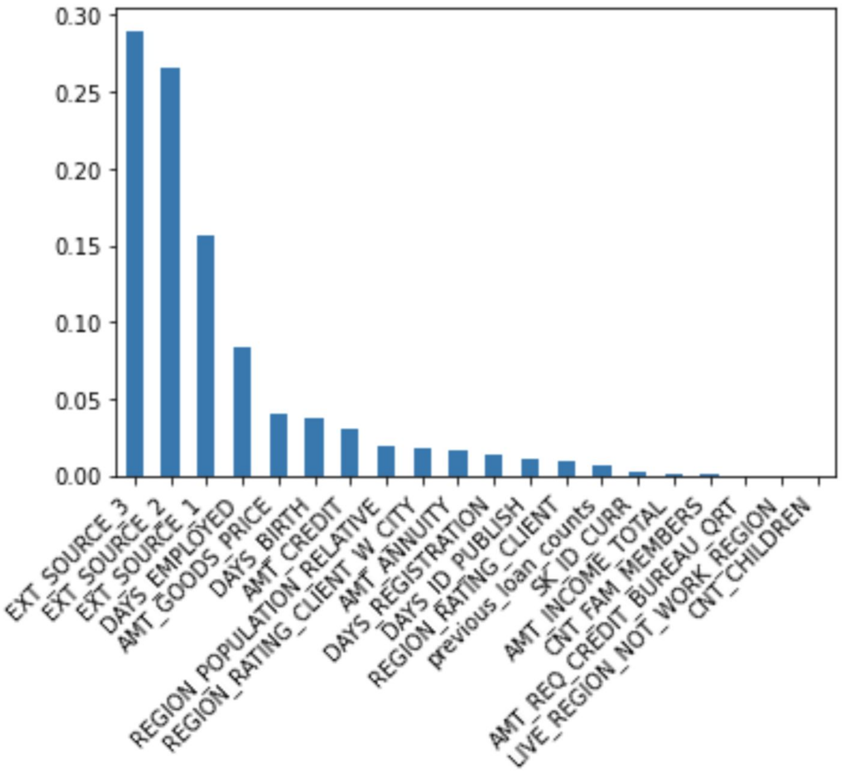

RandomForest (AP = 0.85)
threshold zero
Precision (Positive label: 1)
Recall (Positive label: 1)
1.0
0.9
0.8
0.7
0.6
0.0
0.2
0.4
0.6
0.8
1.0


643
270
167
758
True label
Predicted label
0
1
0
1
600
400
200

Random Forest Feature Importance

Calculated with

feature_importances_



Summary of Findings

Conclusion / Recommendations

What’re the results of the models?

Recall: 68% for logistic regression model, 82% for random forest

Attributes associated with loan repayment: price of goods for which the loan is given, days employed, credit amount, age

Loaning to applicants who banks deem at risk of defaulting is socially productive, yet extreme care must be taken in identifying those who should receive loans.

Thank you!