To loan or not to loan – That is the question

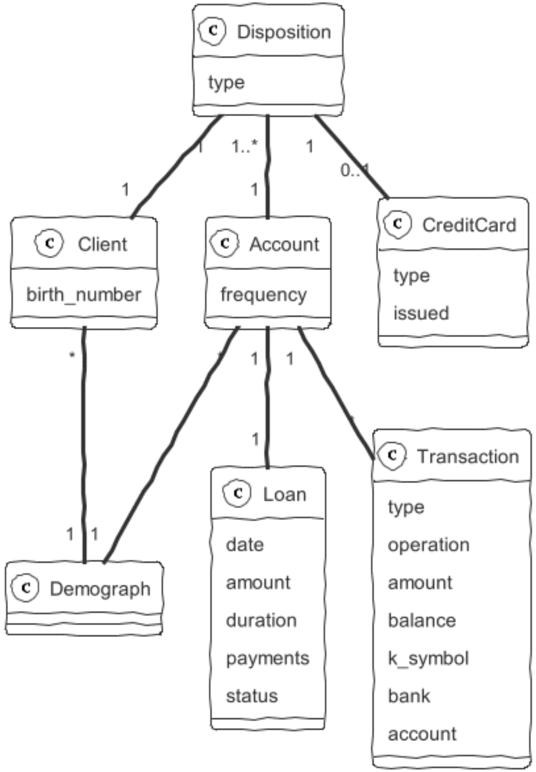
Practical assignment of the *Machine Learning* course at U.Porto

Group

Ricardo Fontão - *up201806317*
João Cardoso - *up201806531*
Eduardo Correia - *up201806433*

Domain Description

- Dataset from the records of a **Czech bank**, dating from **1993** up to **1998**.
- Data was provided in **.csv** files, containing information about accounts, transactions, district information, previous loans and their result, *etc*… each in a separate file



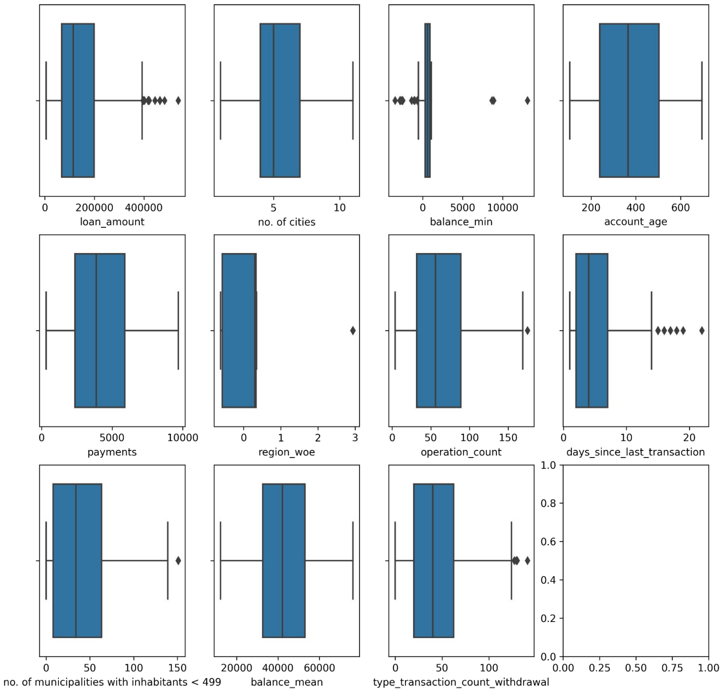
Exploratory Data Analysis

Outlier detection

- (Box) plot each non-categorical field; found columns with possible outliers.
- Found those outliers using standard deviation and first/third quartile methods.
- Analyzed them manually and test with/without them (removing outliers would've worsened results)

Null-values

- Only 11 clients who had loans also had a (credit) card. The card table was dropped.
- The transaction table had very few (not null) k_symbol, bank and account values. Those columns were dropped.
- The district table had a handful of null values on 2 columns. They were replaced with that column's median.



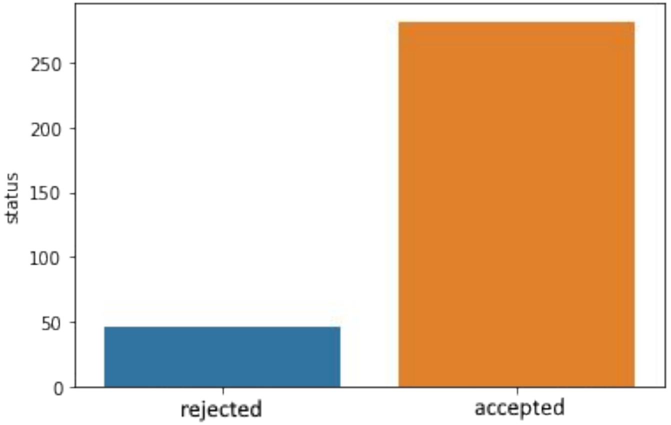
Exploratory Data Analysis

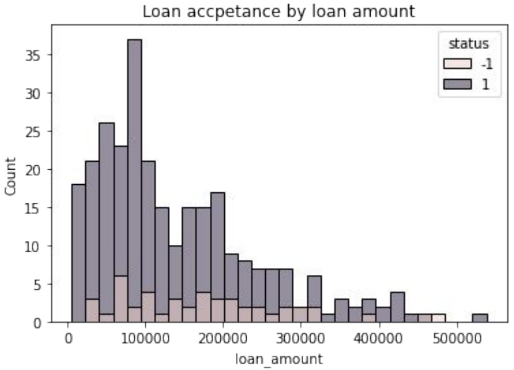
Interesting observations

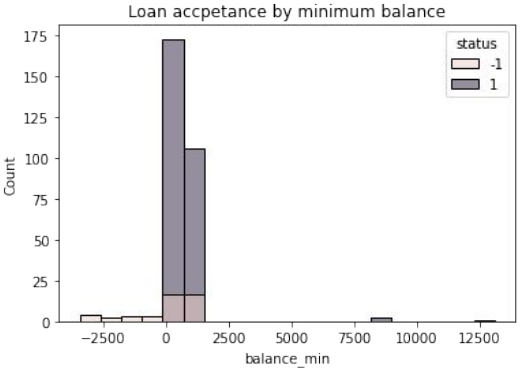
- Almost every client that has had a **balance below 0** had their loan **rejected**.
- There is a negligible correlation between **age** and **sex** with **status**.
- The same client was never in multiple accounts
- Accounts only have 1 or 2 clients each
- Accounts with 2 clients are significantly more likely to be accepted
- The dataset is unbalanced:
    - It has very little refused loans (<50).

Expected observations

- The loan amounts graph tends towards the left half of the plot
- There were no clients requesting loans with ages below 15 or above 65







Data Quality

After carefully analyzing the data, we concluded that it doesn't fulfill any of the data quality dimensions

Completeness

- There are many missing values and no data from the *permanent order* relation

Consistency

- *Operation* field contains stray "withdrawal in cash" value

Conformity

- *birth_number* gives both birthdate and sex information

Accuracy

- Client sex doesn't affect loan status, unlike the real world, where sex plays a major role in loan application

Integrity

- There are client relation entries with no correspondent loans

Timeliness

- Provided dataset is already over two decades old

Problem Definition

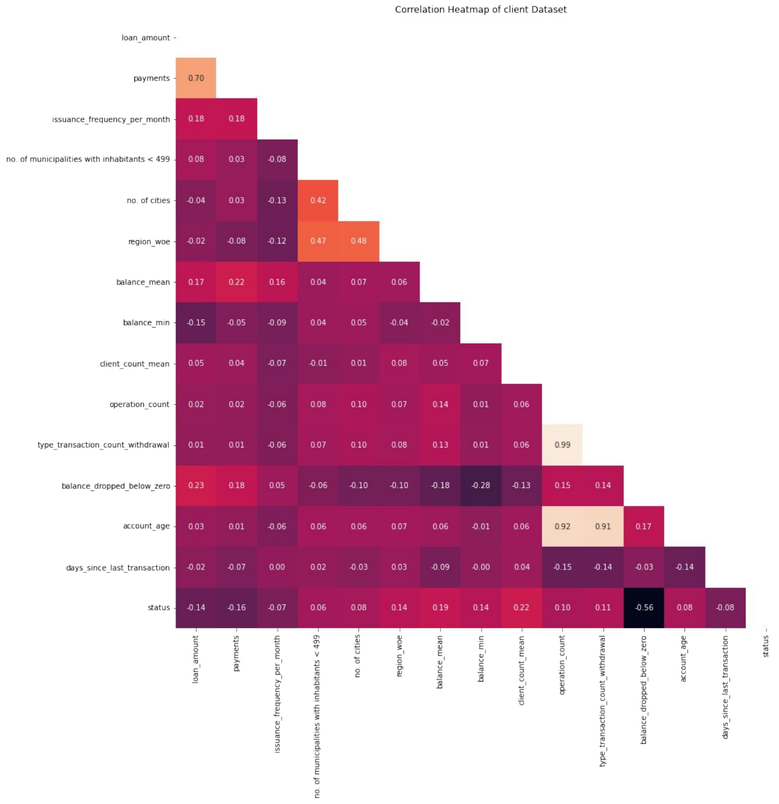
- **Objective:** Predict the probability of a loan not succeeding, given past loans (paid **1**, or unpaid **-1**) and client information.
- **Procedure:** Train a set of machine learning models and compare their performances to achieve the best result possible.
- **Evaluation metric:** Area Under ROC Curve (**AUC**). It’s also better to predict not giving someone a loan because in that case the bank doesn’t lose money

Data Preparation

- Converted dates to “YYYY/MM/DD” format
- Categorized *sex* field
- Encoded region name, using *weight of evidence* encoding (single column encoding whose results are the ln(%status=1/%status=-1))
- Converted categorical values into dummy variables in transaction *operation* field
- Filled missing values in *unemployment rate ‘95* and *no. of committed crimes ‘95* fields using median values
- Detected outliers (none were found)
- Dropped features with high-correlation (for example, *payment* and *duration*, since these are a function of the *amount* field)
- Converted "withdrawal in cash" to "withdrawal" in transaction’s *type* field
- Dropped all *card* data, since there were only 11 cards associated with accounts
- Dropped *unemployment. rate ‘95*, *unemployment. rate ‘96*, *no. crimes ‘95*, *no. crimes ‘96* and *no. entrepreneurs per 1000 habitants*
- Removed null values and features with low impact

Feature Engineering

- Created a field for the number of clients of a given account
- Extracted *min*, *max* and *mean* aggregations for *balance* and *amount* fields
- Created a field to indicate whether the transaction *balance* of a given loan has reached negative values or not
- Unified account issuance frequency by a month period
- Created fields *criminality_growth*, *unemployment_growth* and *ratio_enterpreneurs*
- Extracted client age (at time of loan) and gender from *birth_number* field
- Used logarithmic transformation for skewed values



Experimental Setup

Our project pipeline is as follows:

- Preprocessing
  - Load and process data (drop values, create new features, extract information…)
  - Aggregate all data into just one table
  - Employ clustering techniques
  - Visualize data with different kinds of charts
- Prediction
  - Tune set models
  - For each, plot respective ROC curves and confusion matrix
  - Fit best model to test data and save the obtained results

Experimental Setup

Models tested

- SVC
- KNN
- Decision Tree
- Random Forest
- Naive Bayes
- Logistic Regression Classifier

Feature Selection

- Single filter method (**Anova Test**)
- Chooses top 10 features

Oversampling

- Use of **SMOTE**

Scaling

- Use of Sklearn's **StandardScaler**

- For each model, Sklearn's **GridSearchCV** was used to search for the best hyperparameters to each model.
- To reduce overfitting, **Cross-Validation** was used
  - 5 folds
- For each model, 4 scenarios were analyzed
  - No Feature Selection and No Oversampling
  - Only Feature Selection
  - Only Oversampling
  - Feature Selection and Oversampling

Results

ROC-AUC
Decision Tree
SVC
K-nearest Neighbours
Naive Bayes
Random Forest
Logistic Regression
0.83 0.82 0.83 0.87
0.88 0.87 0.87 0.87
0.84 0.86 0.86 0.87
0.87 0.85 0.88 0.85
0.88 0.88 0.88 0.88
0.86 0.87 0.86 0.88
No Feature selection/No oversampling
Feature Selection
Oversampling
Feature Selection/Oversampling

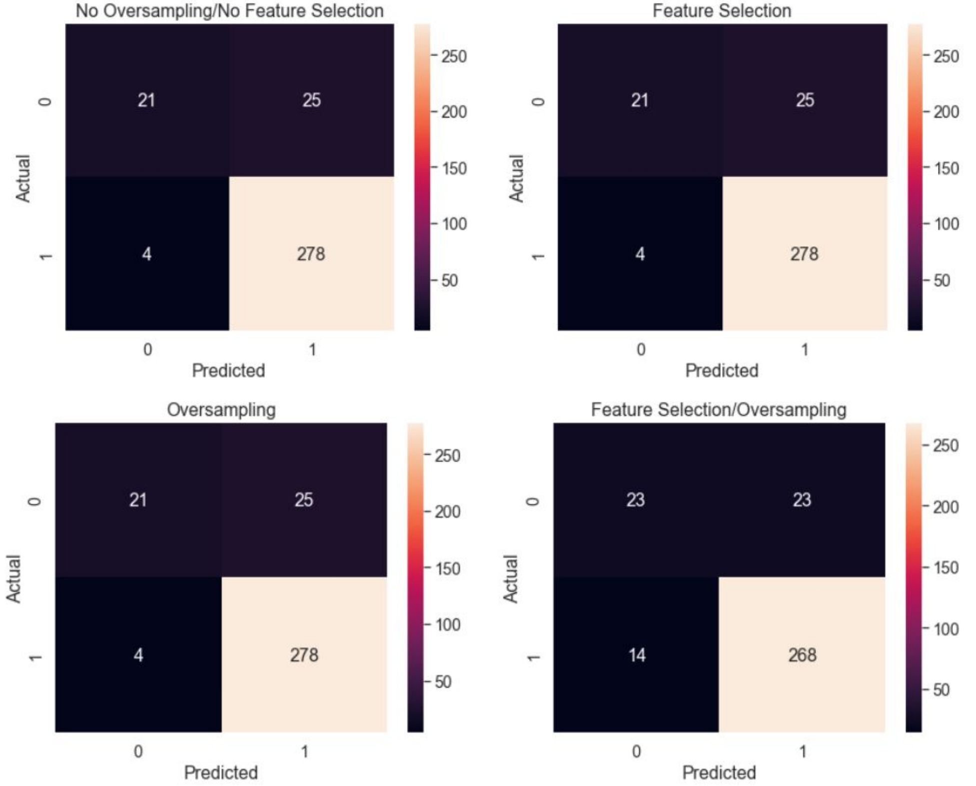
Accuracy
Decision Tree
SVC
K-nearest Neighbours
Naive Bayes
Random Forest
Logistic Regression
0.93 0.86 0.91 0.86
0.7 0.91 0.89 0.89
0.86 0.92 0.77 0.82
0.86 0.86 0.86 0.86
0.93 0.91 0.9 0.9
0.72 0.84 0.83 0.83
No Feature selection/No oversampling
Feature Selection
Oversampling
Feature Selection/Oversampling

F1-Score
Decision Tree
SVC
K-nearest Neighbours
Naive Bayes
Random Forest
Logistic Regression
0.96 0.92 0.95 0.92
0.8 0.95 0.94 0.94
0.92 0.95 0.84 0.88
0.92 0.92 0.92 0.92
0.96 0.95 0.94 0.94
0.82 0.9 0.89 0.89
No Feature selection/No oversampling
Feature Selection
Oversampling
Feature Selection/Oversampling

Results

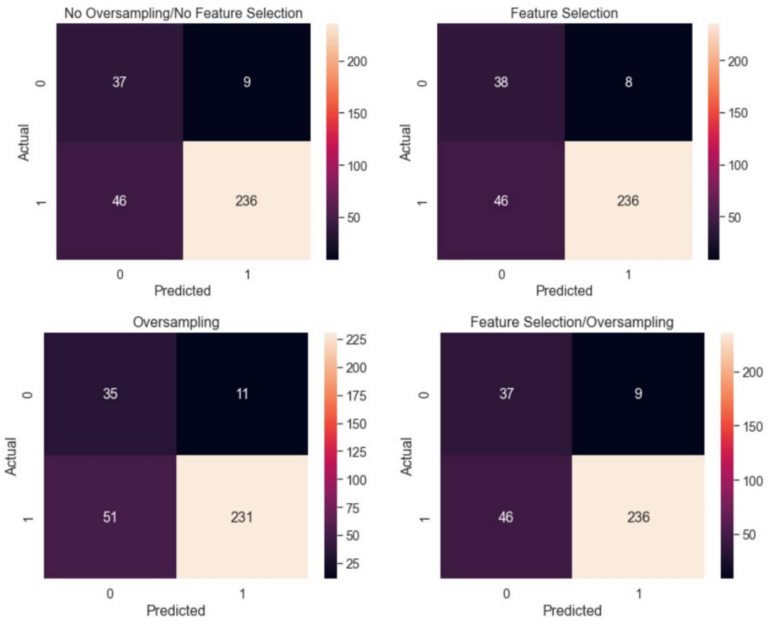
- All classifiers presented similar results
- The one that stands out the most in terms of metrics is the **Random Forest Classifier**

- Both **Feature Selection** and **Oversampling** don’t appear to influence the results very much

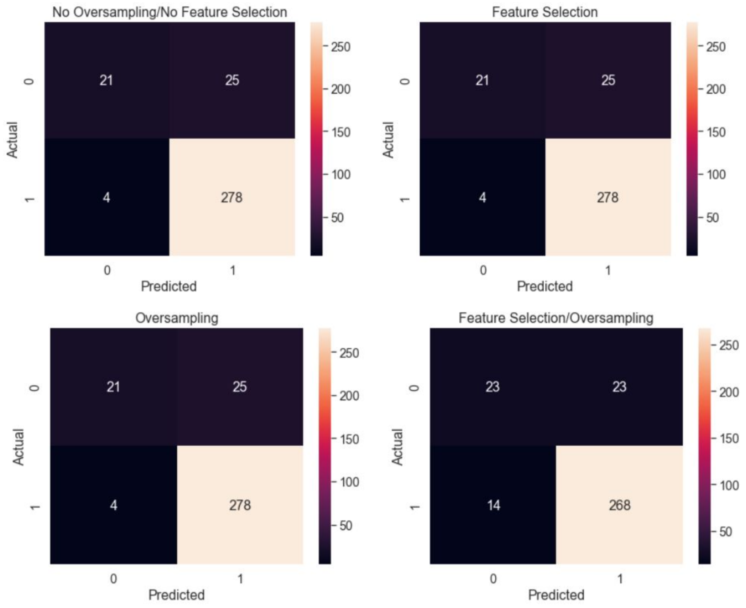


Logistic Regression vs Random Forest

Logistic Regression



Random Forest

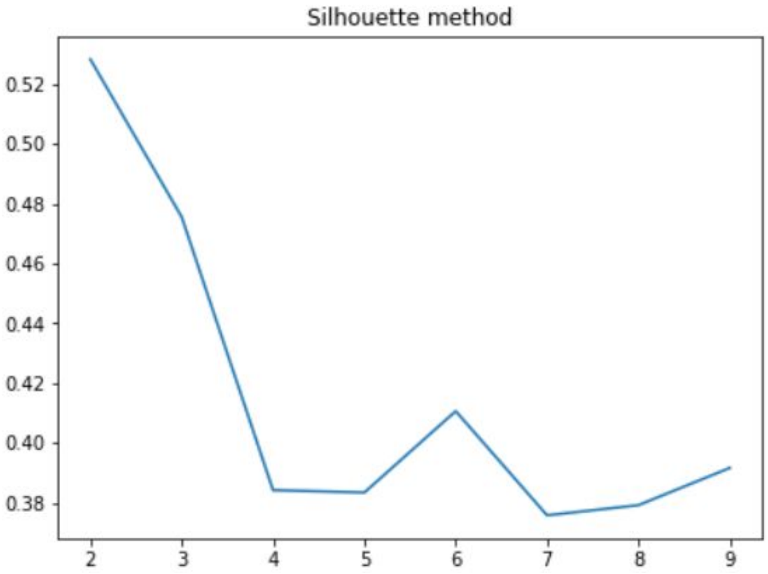


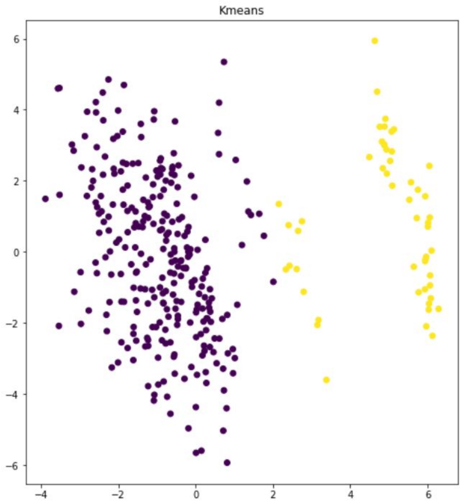
Descriptive Analysis

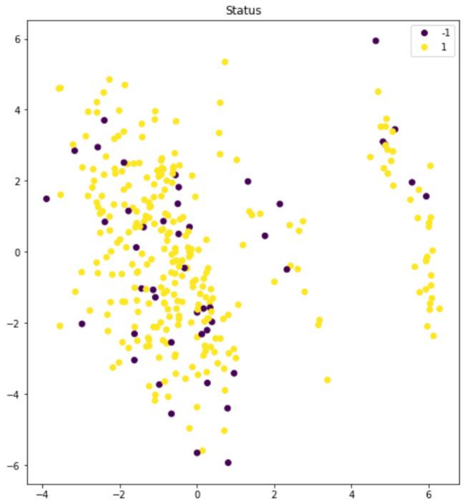
- PCA was used to reduce the dimensionality of the data
- The silhouette method was used to choose the number of clusters
- Kmeans was the clustering method used

All features

From using the silhouette method, we can infer that 2 clusters is the best starting amount. However, when plotting the clusters, no clear aggregations could be visualized. When plotting which points had status 1 or -1 there still were no conclusions regarding groups of costumers.

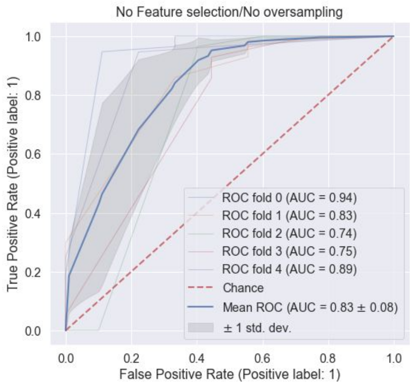


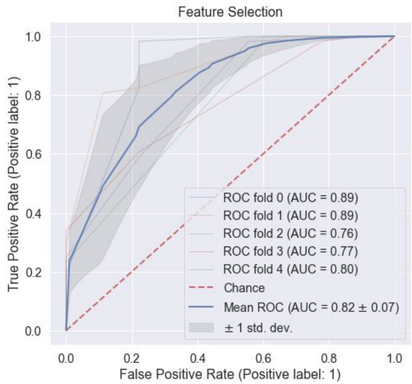


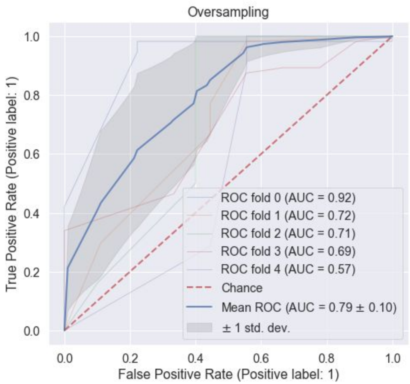


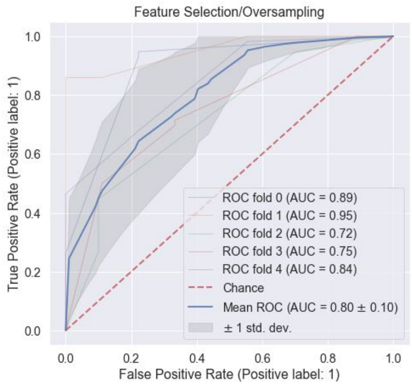
Models

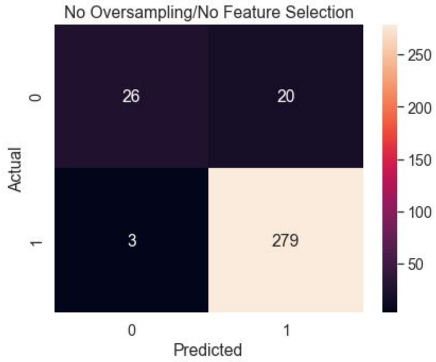
Decision Tree

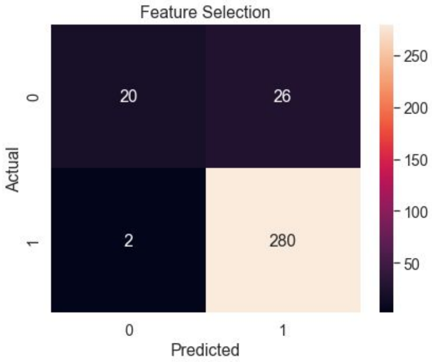


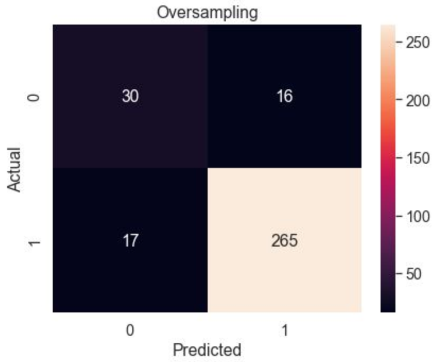


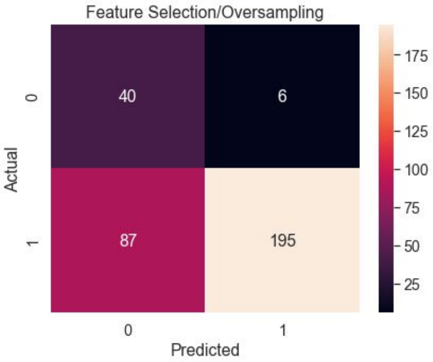




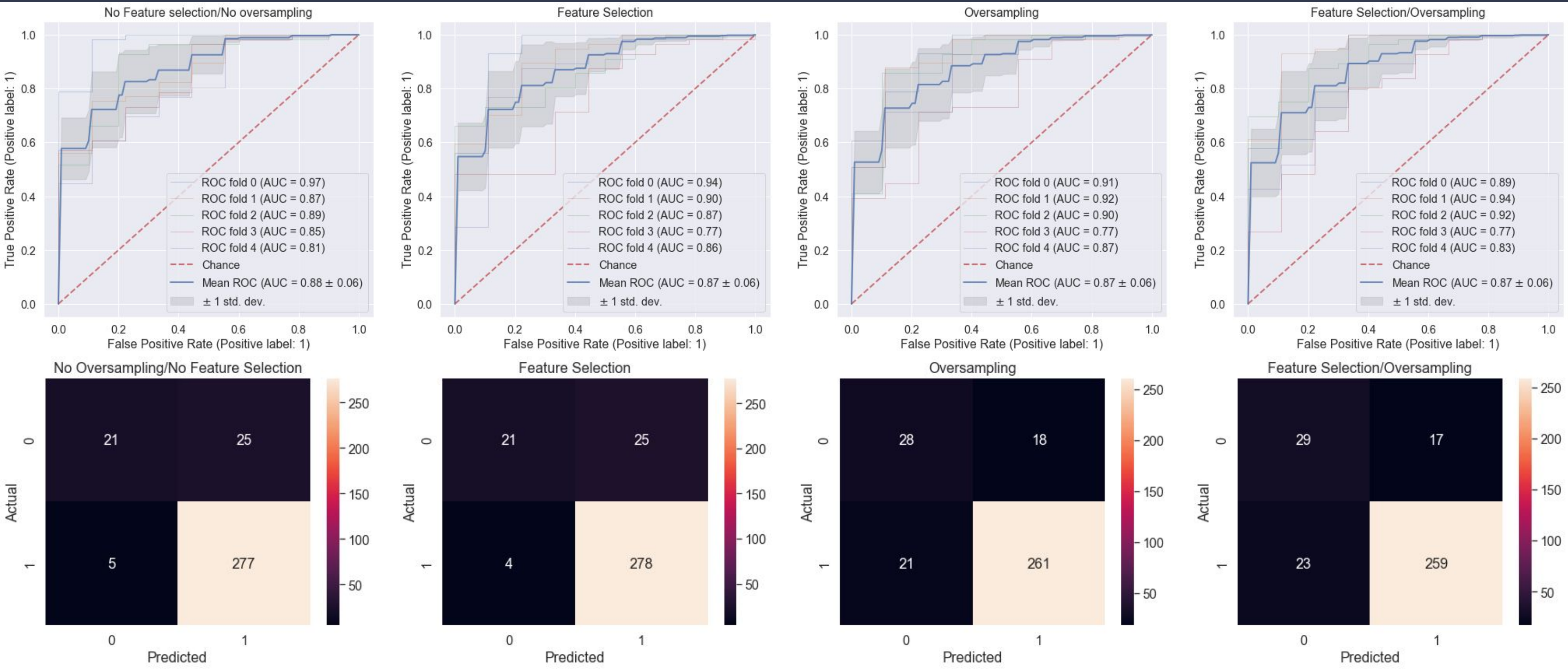




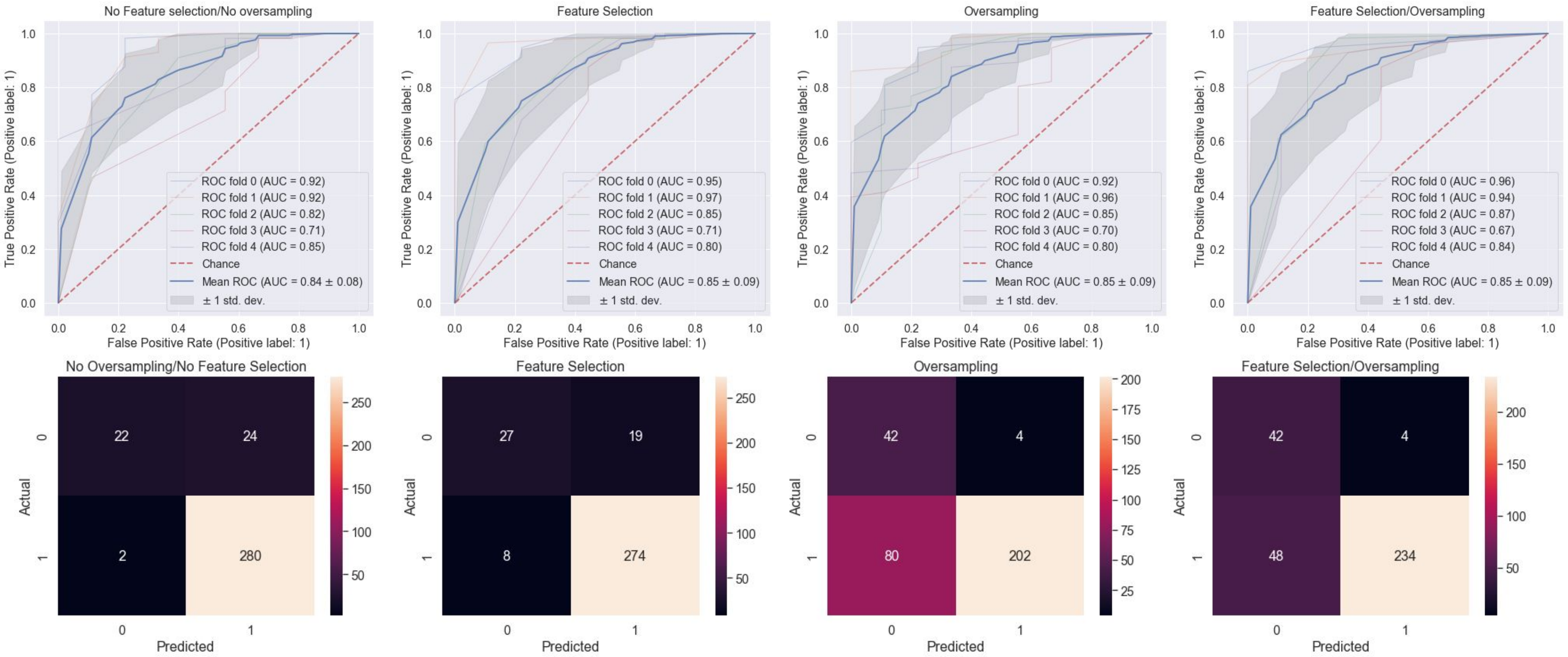




SVM

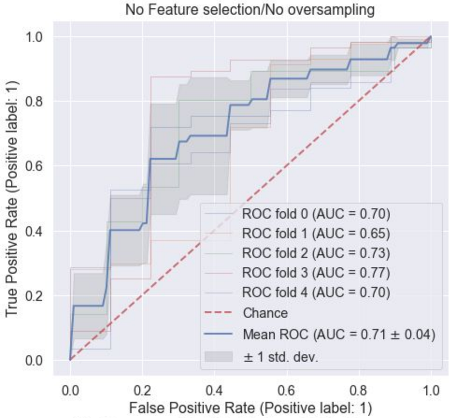

No Feature selection/No oversampling
ROC fold 0 (AUC = 0.97)
ROC fold 1 (AUC = 0.87)
ROC fold 2 (AUC = 0.89)
ROC fold 3 (AUC = 0.85)
ROC fold 4 (AUC = 0.81)
Chance
Mean ROC (AUC = 0.88 ± 0.06)
± 1 std. dev.
Feature Selection
ROC fold 0 (AUC = 0.94)
ROC fold 1 (AUC = 0.90)
ROC fold 2 (AUC = 0.87)
ROC fold 3 (AUC = 0.77)
ROC fold 4 (AUC = 0.86)
Chance
Mean ROC (AUC = 0.87 ± 0.06)
± 1 std. dev.
Oversampling
ROC fold 0 (AUC = 0.91)
ROC fold 1 (AUC = 0.92)
ROC fold 2 (AUC = 0.90)
ROC fold 3 (AUC = 0.77)
ROC fold 4 (AUC = 0.87)
Chance
Mean ROC (AUC = 0.87 ± 0.06)
± 1 std. dev.
Feature Selection/Oversampling
ROC fold 0 (AUC = 0.89)
ROC fold 1 (AUC = 0.94)
ROC fold 2 (AUC = 0.92)
ROC fold 3 (AUC = 0.77)
ROC fold 4 (AUC = 0.83)
Chance
Mean ROC (AUC = 0.87 ± 0.06)
± 1 std. dev.
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
No Oversampling/No Feature Selection
21
25
5
277
Feature Selection
21
25
4
278
Oversampling
28
18
21
261
Feature Selection/Oversampling
29
17
23
259
Actual
Predicted
0
1
50
100
150
200
250

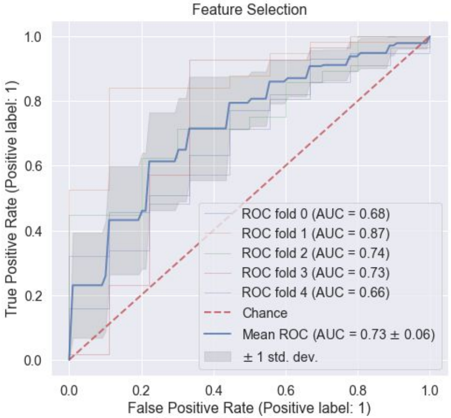
KNN

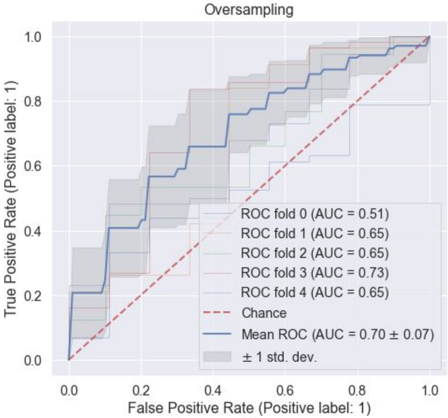


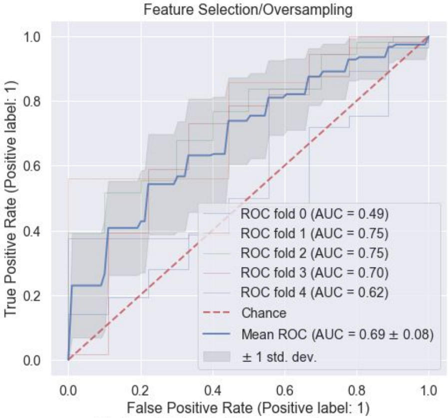
No Feature selection/No oversampling
ROC fold 0 (AUC = 0.92)
ROC fold 1 (AUC = 0.92)
ROC fold 2 (AUC = 0.82)
ROC fold 3 (AUC = 0.71)
ROC fold 4 (AUC = 0.85)
Chance
Mean ROC (AUC = 0.84 ± 0.08)
± 1 std. dev.
Feature Selection
ROC fold 0 (AUC = 0.95)
ROC fold 1 (AUC = 0.97)
ROC fold 2 (AUC = 0.85)
ROC fold 3 (AUC = 0.71)
ROC fold 4 (AUC = 0.80)
Chance
Mean ROC (AUC = 0.85 ± 0.09)
± 1 std. dev.
Oversampling
ROC fold 0 (AUC = 0.92)
ROC fold 1 (AUC = 0.96)
ROC fold 2 (AUC = 0.85)
ROC fold 3 (AUC = 0.70)
ROC fold 4 (AUC = 0.80)
Chance
Mean ROC (AUC = 0.85 ± 0.09)
± 1 std. dev.
Feature Selection/Oversampling
ROC fold 0 (AUC = 0.96)
ROC fold 1 (AUC = 0.94)
ROC fold 2 (AUC = 0.87)
ROC fold 3 (AUC = 0.67)
ROC fold 4 (AUC = 0.84)
Chance
Mean ROC (AUC = 0.85 ± 0.09)
± 1 std. dev.
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
No Oversampling/No Feature Selection
22
24
2
280
Feature Selection
27
19
8
274
Oversampling
42
4
80
202
Feature Selection/Oversampling
42
4
48
234
Actual
Predicted

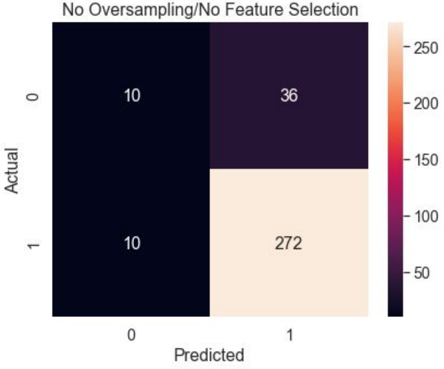
Naive Bayes

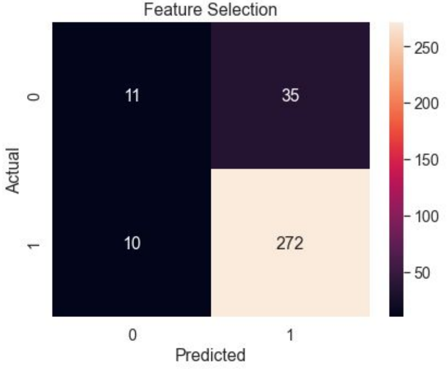

No Feature selection/No oversampling
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
ROC fold 0 (AUC = 0.70)
ROC fold 1 (AUC = 0.65)
ROC fold 2 (AUC = 0.73)
ROC fold 3 (AUC = 0.77)
ROC fold 4 (AUC = 0.70)
Chance
Mean ROC (AUC = 0.71 ± 0.04)
± 1 std. dev.

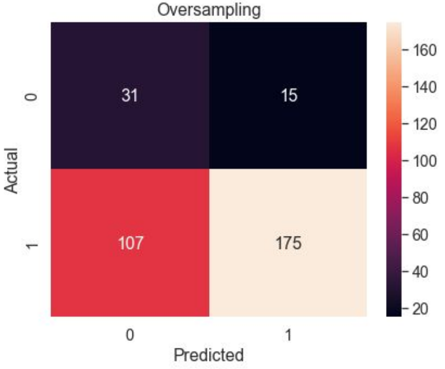

Feature Selection
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
ROC fold 0 (AUC = 0.68)
ROC fold 1 (AUC = 0.87)
ROC fold 2 (AUC = 0.74)
ROC fold 3 (AUC = 0.73)
ROC fold 4 (AUC = 0.66)
Chance
Mean ROC (AUC = 0.73 ± 0.06)
± 1 std. dev.

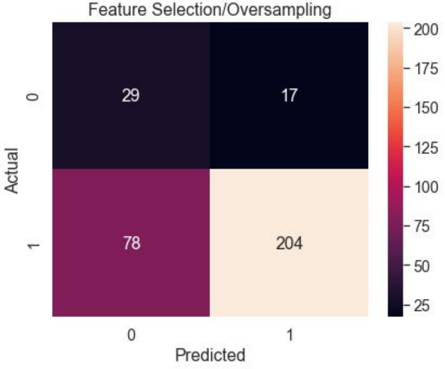

Oversampling
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
ROC fold 0 (AUC = 0.51)
ROC fold 1 (AUC = 0.65)
ROC fold 2 (AUC = 0.65)
ROC fold 3 (AUC = 0.73)
ROC fold 4 (AUC = 0.65)
Chance
Mean ROC (AUC = 0.70 ± 0.07)
± 1 std. dev.


Feature Selection/Oversampling
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
ROC fold 0 (AUC = 0.49)
ROC fold 1 (AUC = 0.75)
ROC fold 2 (AUC = 0.75)
ROC fold 3 (AUC = 0.70)
ROC fold 4 (AUC = 0.62)
Chance
Mean ROC (AUC = 0.69 ± 0.08)
± 1 std. dev.

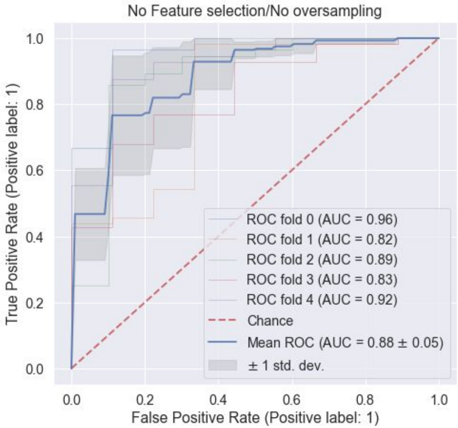

No Oversampling/No Feature Selection
10
36
10
272
Actual
Predicted

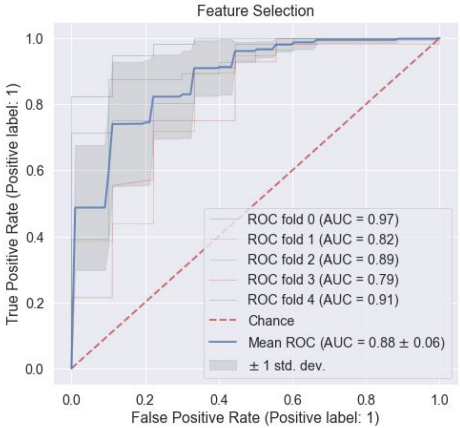

Feature Selection
11
35
10
272
Actual
Predicted

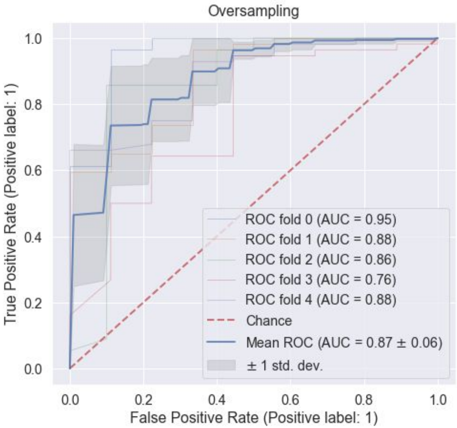

Oversampling
31
15
107
175
Actual
Predicted

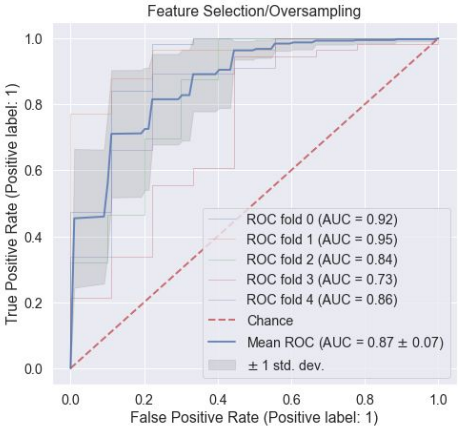

Feature Selection/Oversampling
29
17
78
204
Actual
Predicted

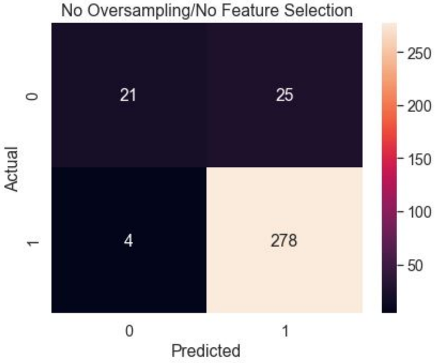
Random Forest

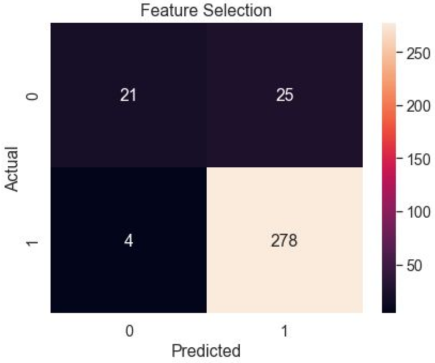


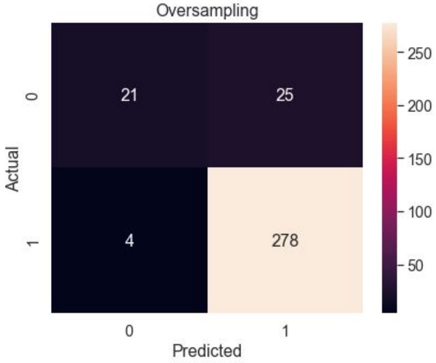


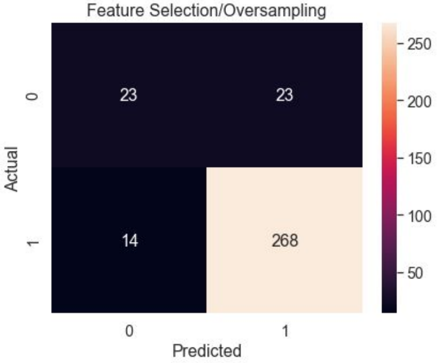




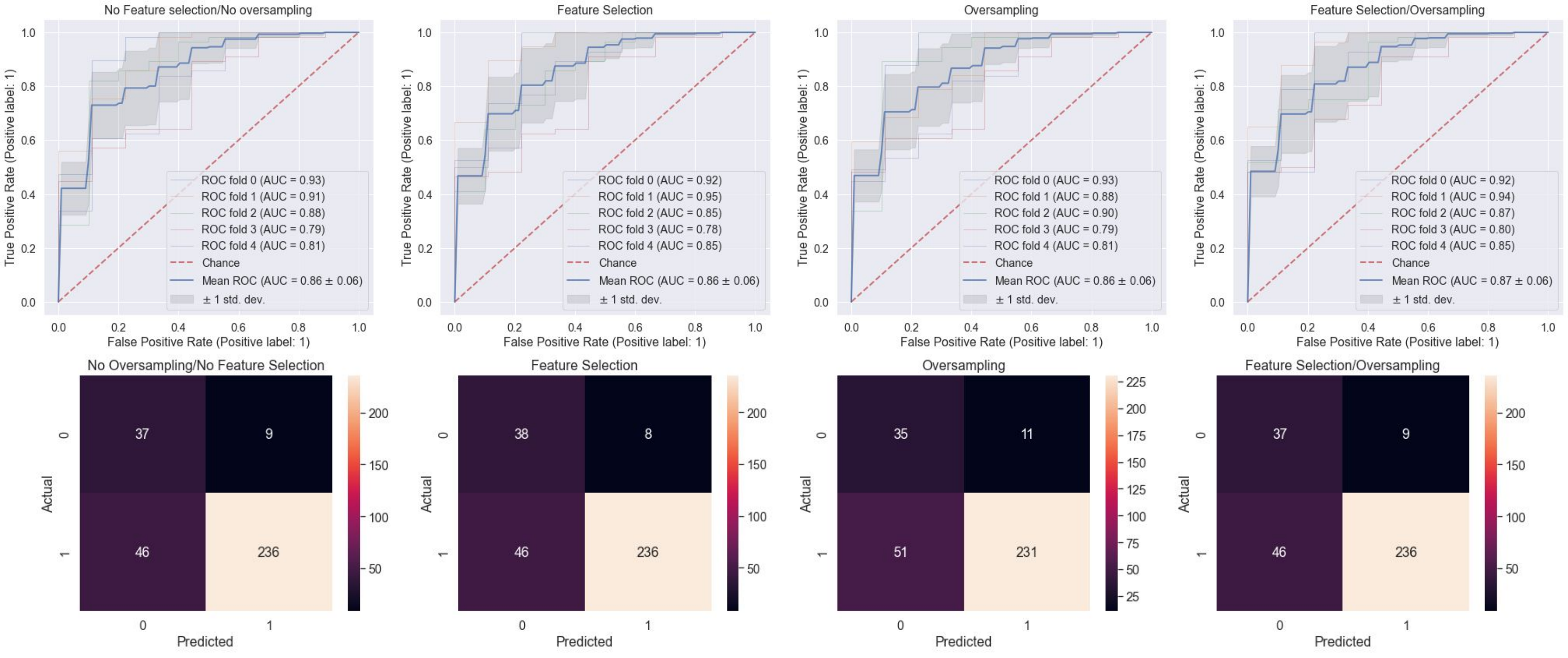








Logistic Regression



Best and worst performing

- Best:
    - Random Forest
    - Decision Tree
    - K nearest neighbors (KNN)
- Worst:
    - Logistic regression
        - However, it defaults to -1, which is good for the bank, because losing money to an accepted loan which doesn’t work is likely worse than not gaining money from a would-be successful loan
    - Naive Bayes

Second Best – Decision Tree

- Model explanation:
  - Start with a root node R; it contains the full dataset.
  - For each node:
    - Select the best feature, create nodes for each of its values containing every part of the dataset.
  - Repeat step 2 until the nodes can't be split, or max depth is reached.
- Positive: Handles numerical and categorical data
- Negative: Relatively prone to overfitting

```
parameter_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': range(1, 7)
}

dt, dt_fs, dt_os, dt_fs_os = (tune_model(
    train_df,
    DecisionTreeClassifier(random_state=RANDOM_STATE),
    parameter_grid,
    columns_to_drop,
    target_column,
    oversample=oversample,
    feature_selection=feature_selection
) for oversample, feature_selection in ((False, False), (False, True), (True, False), (True,True)))
```

https://towardsmachinelearning.org/decision-tree-algorithm/

Best – Random Forest

- Model explanation:
  - Generate a number of decision trees, each with a different sample of the training dataset.
  - (See Decision Tree)
  - For each value in the testing dataset, run reach decision tree and the result (status value) with the most votes is chosen.

Positive: More resistant to overfitting than random forest.

```
parameter_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [5, 10, 15],
    'n_jobs': [-1],  # Use all cores
    'criterion': ['gini', 'entropy']
}

rfc, rfc_fs, rfc_os, rfc_fs_os = (tune_model(
    train_df,
    RandomForestClassifier(random_state=RANDOM_STATE),
    parameter_grid,
    columns_to_drop,
    target_column,
    oversample=oversample,
    feature_selection=feature_selection
) for oversample, feature_selection in ((False, False), (False, True), (True, False), (True,True)))
```

https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/#how-does-the-random-forest-algorithm-work

Third Best – K nearest neighbors (KNN)

- Model explanation:
  - Select a number of neighbors K for any new point:
  - Take the K nearest neighbors (based on Euclidean distance)
  - Count the number of points for each category in those K
  - The new point is assumed to belong to the category with the highest amount.

```
parameter_grid = {
    'n_neighbors': [4, 5, 6, 7, 10, 15],
    'leaf_size': [5, 10, 15, 20, 50, 100],
    'n_jobs': [-1],
    'algorithm': ['auto']
}

knn, knn_fs, knn_os, knn_fs_os = (tune_model(
    train_df,
    neighbors.KNeighborsClassifier(),
    parameter_grid,
    columns_to_drop,
    target_column,
    scaler=StandardScaler(),
    oversample=oversample,
    feature_selection=feature_selection
) for oversample, feature_selection in ((False, False), (False, True), (True, False), (True,True)))
```

Positives:

- Robust to noisy data

Negatives:

- Need to find best K
- Best with high amount of data (not our case)

https://people.revoledu.com/kardi/tutorial/KNN/Strength%20and%20Weakness.htm

Worst – Naive Bayes

Model explanation:

- Calculate the probability of success (status = 1) for each value of each feature (P(Y|X))
- For each value, multiply each of its features' values' probabilities to obtain the probability of success.

```
nb, nb_fs, nb_os, nb_fs_os = (tune_model(
    train_df,
    GaussianNB(),
    parameter_grid,
    columns_to_drop,
    target_column,
    scaler=StandardScaler(),
    oversample=oversample,
    feature_selection=feature_selection
) for oversample, feature_selection in ((False, False), (False, True), (True, False), (True,True)))
```

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \rightarrow Posterior = \frac{likelihood \times prior}{evidence}$$

Positives:

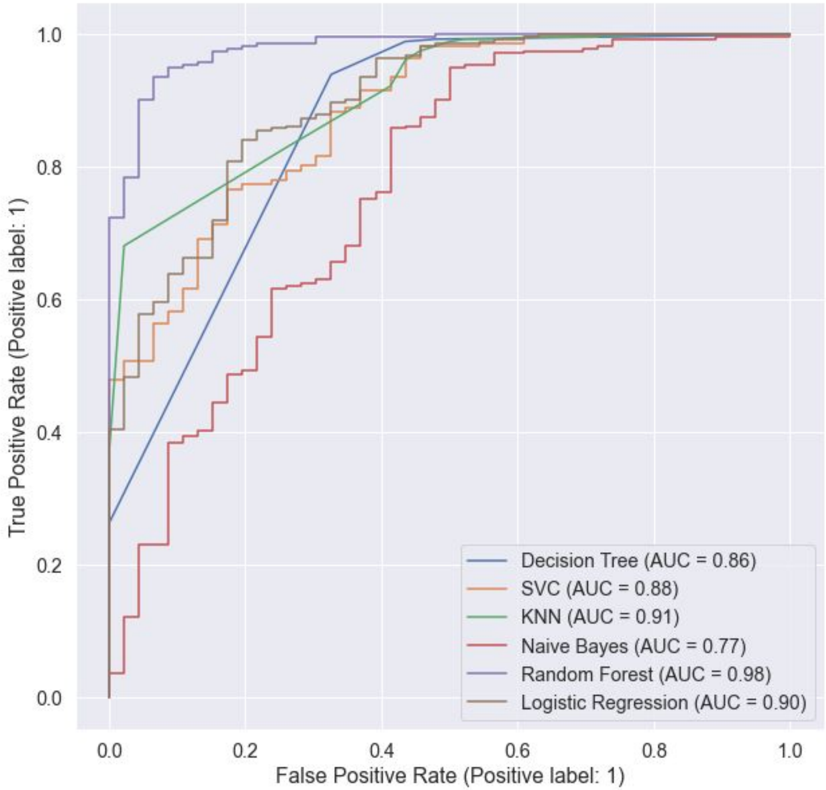
- Fast (handles high dimensional data easily)
- Less bad with little data

Negatives:

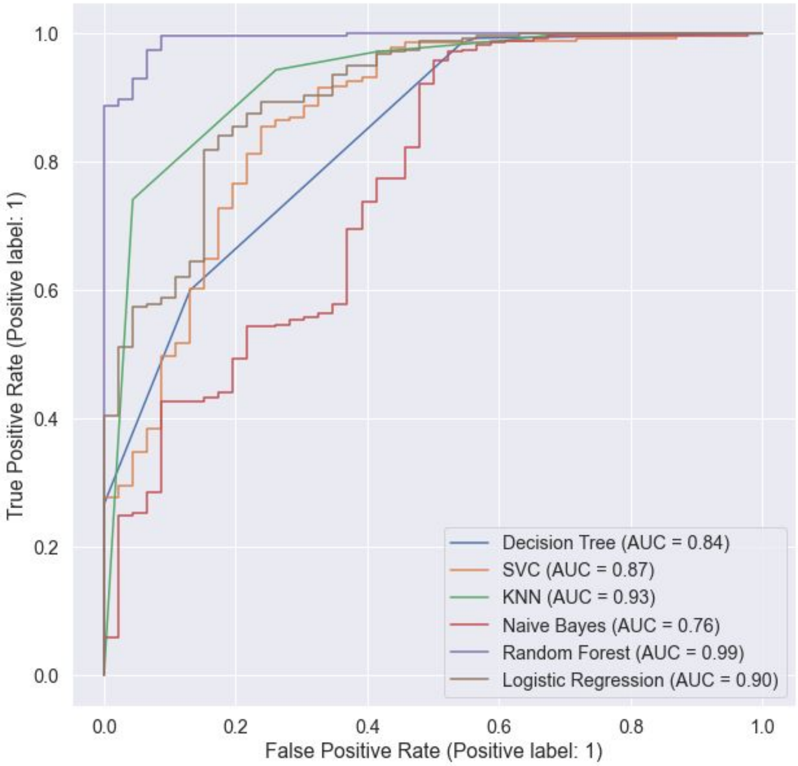
- Assumes each variable is independent (gives bad results if there are many variables with high correlations)
- Bad for probability estimation

https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0

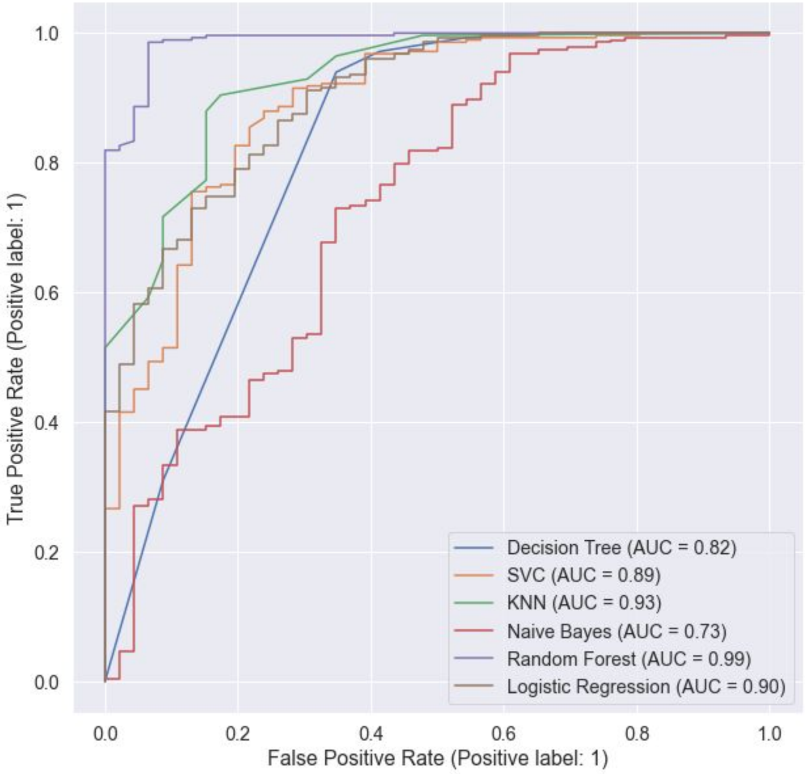
No Feature Selection / No oversampling



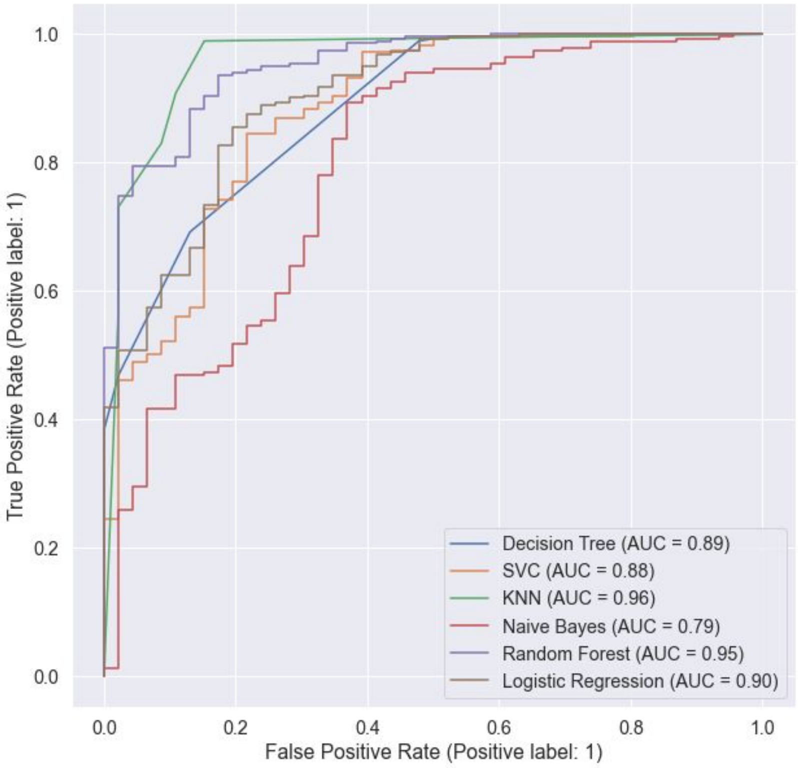
Feature Selection / No Oversampling



No Feature Selection / With Oversampling



Feature Selection / Oversampling



Conclusion

- Some algorithms could not be used effectively due to the fact that the available data was so scarce (for instance, *DeepLearning*).
- Some data (for instance, the *card* relation) proved to be irrelevant.
- There were missing fields and even relations (for instance, the *permanent order* relation) from the banking case description.
- Tree-based models perform well, but tend to overfit if not configured correctly and with many features.
- Filter Methods for *feature selection* and *oversampling* didn't improve or worsen the results

Future Work

- Experiment with new models (for instance, *VotingClassifier*, to take advantage of our best models)
- Test more feature selection algorithms
- Further improve features being used (with feature engineering/selection)
- Apply other oversampling techniques other than SMOTE and also undersampling

Used Tools

- **Python/Jupyter Notebook:** Development of the entire project
  - *numpy* - Numerical handling of matrix-like data
  - *pandas* - Data manipulation
  - *imblearn* - Dealing with imbalanced classes
  - *sklearn* - Classifier algorithms
  - *matplotib/seaborn* - Data visualization

Self-Evaluation

Ricardo Fontão - 1

João Cardoso - 1

Eduardo Correia - 1

All students contributed the same amount of effort to the project