



Project 4

Credit Card Loans

Ally Blitch
Shauntel Phillips
Uzor Francis
Fabi Estrada



Introduction

- We chose a dataset on credit card loan classification because we were curious how different demographics affect loan risk.
- We picked this specific dataset from Kaggle because it was already fairly clean and it has a high usability rating.
- Our target is loan risk with 1 being high risk and 0 being low risk



Predictions

Education level and income will be the highest predictors of risk.

Marital status and gender will be the least effective in predicting risk.



Data Engineering

We dropped “phone”, “email”, and “id” columns because they did not have any effect on the outcomes.

We also updated “Gender”, “Own_car”, “Own_property”, “Unemployed”, and “Target” columns to either 0 or 1 values, or “yes” or “no”.

Data Engineering cont.

id	Num_family	Account_length	Total_income	Age	Years_employed	Income_type	Education_type	Family_status	Housing_type	Occupation_type	Target
0	2	15	427500.0	32.868574	12.435574	Working	Higher education	Civil marriage	Rented apartment	Other	1
0	2	29	112500.0	58.793815	3.104787	Working	Secondary / secondary special	Married	House / apartment	Security staff	0
0	1	4	270000.0	52.321403	8.353354	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	Sales staff	0
0	1	20	283500.0	61.504343	0.000000	Pensioner	Higher education	Separated	House / apartment	Other	0
0	2	5	270000.0	46.193967	2.105450	Working	Higher education	Married	House / apartment	Accountants	0
id	Num_family	Account_length	Total_income	Age	Years_employed	Income_type	Education_type	Family_status	Housing_type	Occupation_type	Target
0	2	15	427500.0	32.868574	12.435574	4	1	0	4	12	1
0	2	29	112500.0	58.793815	3.104787	4	4	1	1	17	0
0	1	4	270000.0	52.321403	8.353354	0	4	3	1	15	0
0	1	20	283500.0	61.504343	0.000000	1	1	2	1	12	0
0	2	5	270000.0	46.193967	2.105450	4	1	1	1	0	0



SMOTE

(Synthetic Minority Oversampling Technique)

SMOTE adjusts for imbalanced data. We found this useful because our models were over predicting as low risk.

```
df2.Target.value_counts()
```

```
Target
0      8426
1      1283
Name: count, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
```

```
print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))



print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

After OverSampling, the shape of train_X: (12638, 43)

After OverSampling, the shape of train_y: (12638,)

After OverSampling, counts of label '1': 6319

After OverSampling, counts of label '0': 6319



Logistic Regression

Test Confusion Matrix:
[[2107 0]
[321 0]]

Test Report:

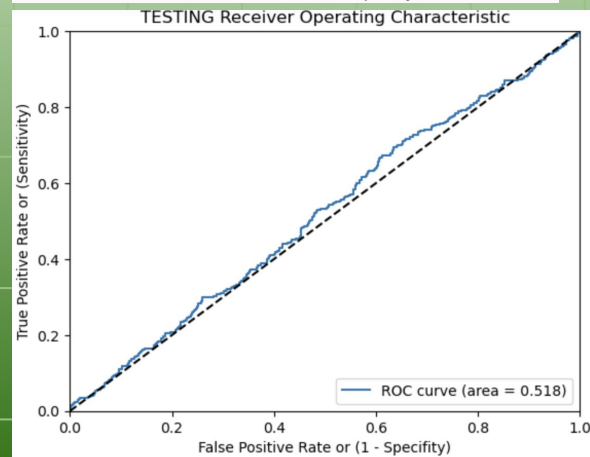
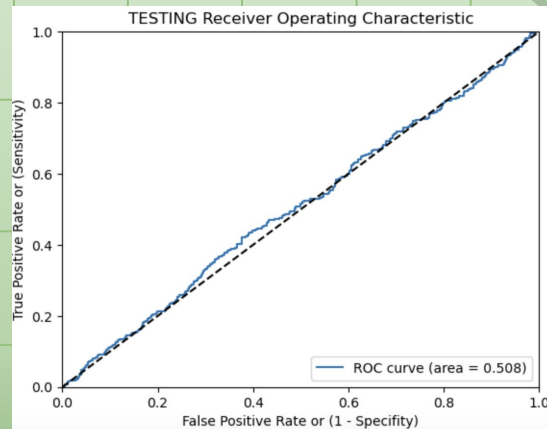
	precision	recall	f1-score	support
0	0.87	1.00	0.93	2107
1	0.00	0.00	0.00	321
accuracy			0.87	2428
macro avg	0.43	0.50	0.46	2428
weighted avg	0.75	0.87	0.81	2428

TESTING METRICS

Test Confusion Matrix:
[[1536 571]
[210 111]]

Test Report:

	precision	recall	f1-score	support
0	0.88	0.73	0.80	2107
1	0.16	0.35	0.22	321
accuracy			0.68	2428
macro avg	0.52	0.54	0.51	2428
weighted avg	0.78	0.68	0.72	2428



Random Forest

Test Confusion Matrix:

```
[[2105  2]
 [ 318  3]]
```

Test Report:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.87	1.00	0.93	2107
---	------	------	------	------

1	0.60	0.01	0.02	321
---	------	------	------	-----

accuracy			0.87	2428
----------	--	--	------	------

macro avg	0.73	0.50	0.47	2428
-----------	------	------	------	------

weighted avg	0.83	0.87	0.81	2428
--------------	------	------	------	------

Test Confusion Matrix:

```
[[2074  33]
 [ 312   9]]
```

Test Report:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.87	0.98	0.92	2107
---	------	------	------	------

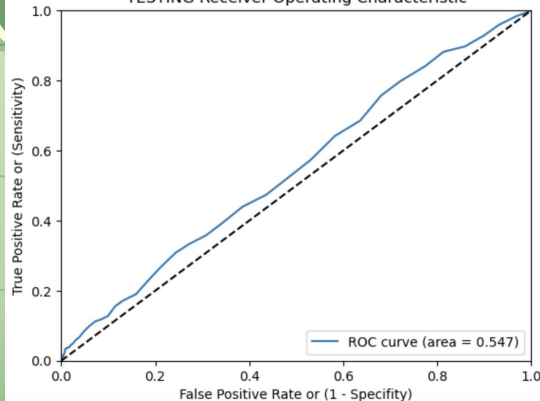
1	0.21	0.03	0.05	321
---	------	------	------	-----

accuracy			0.86	2428
----------	--	--	------	------

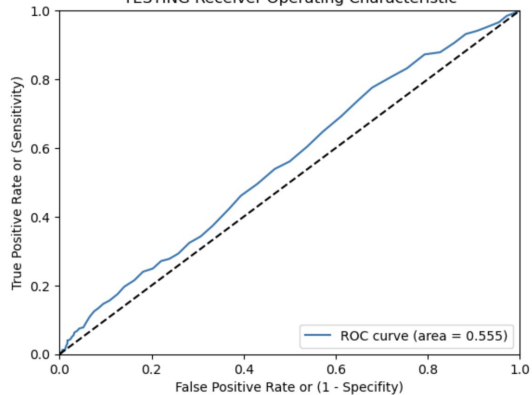
macro avg	0.54	0.51	0.49	2428
-----------	------	------	------	------

weighted avg	0.78	0.86	0.81	2428
--------------	------	------	------	------

TESTING Receiver Operating Characteristic



TESTING Receiver Operating Characteristic



SVC

Test Confusion Matrix:
[[2107 0]
[321 0]]

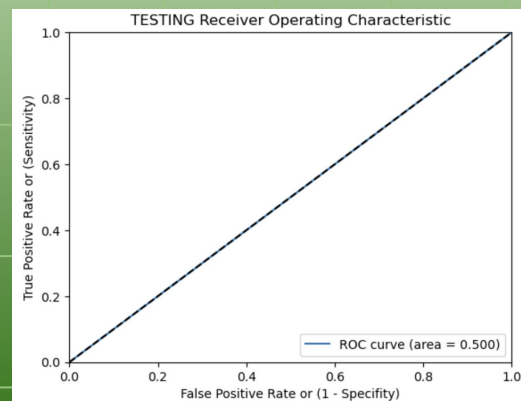
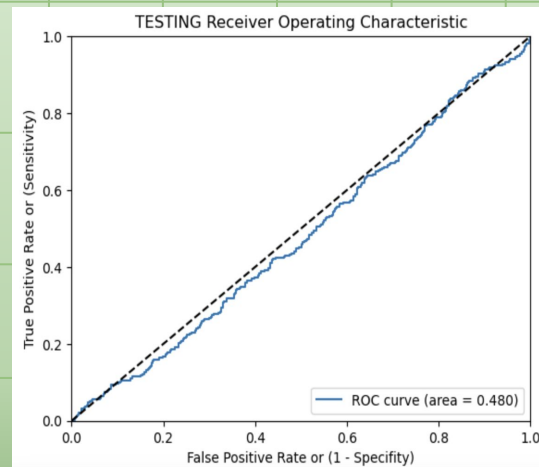
Test Report:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	2107
1	0.00	0.00	0.00	321
accuracy			0.87	2428
macro avg	0.43	0.50	0.46	2428
weighted avg	0.75	0.87	0.81	2428

Test Confusion Matrix:
[[802 1305]
[106 215]]

Test Report:

	precision	recall	f1-score	support
0	0.88	0.38	0.53	2107
1	0.14	0.67	0.23	321
accuracy			0.42	2428
macro avg	0.51	0.53	0.38	2428
weighted avg	0.79	0.42	0.49	2428



KNeighbors

Test Confusion Matrix:

```
[[2086  21]
 [ 318   3]]
```

Test Report:

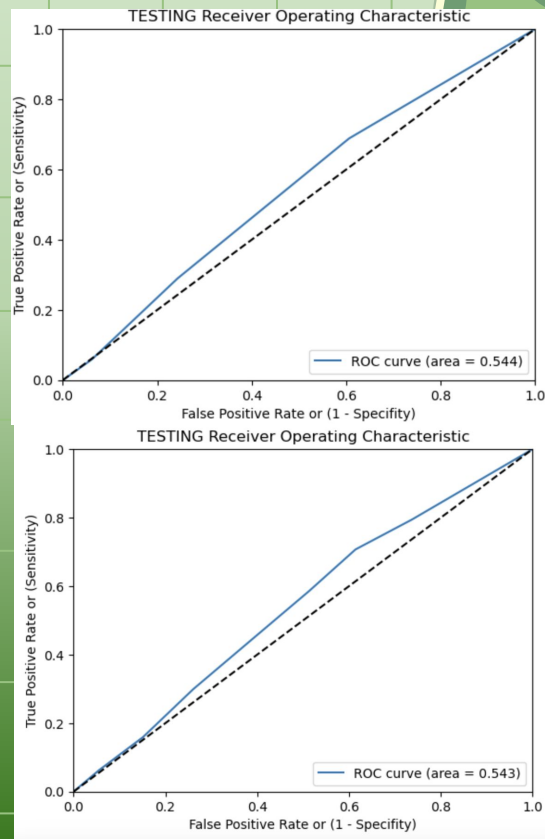
	precision	recall	f1-score	support
0	0.87	0.99	0.92	2107
1	0.12	0.01	0.02	321
accuracy			0.86	2428
macro avg	0.50	0.50	0.47	2428
weighted avg	0.77	0.86	0.80	2428

Test Confusion Matrix:

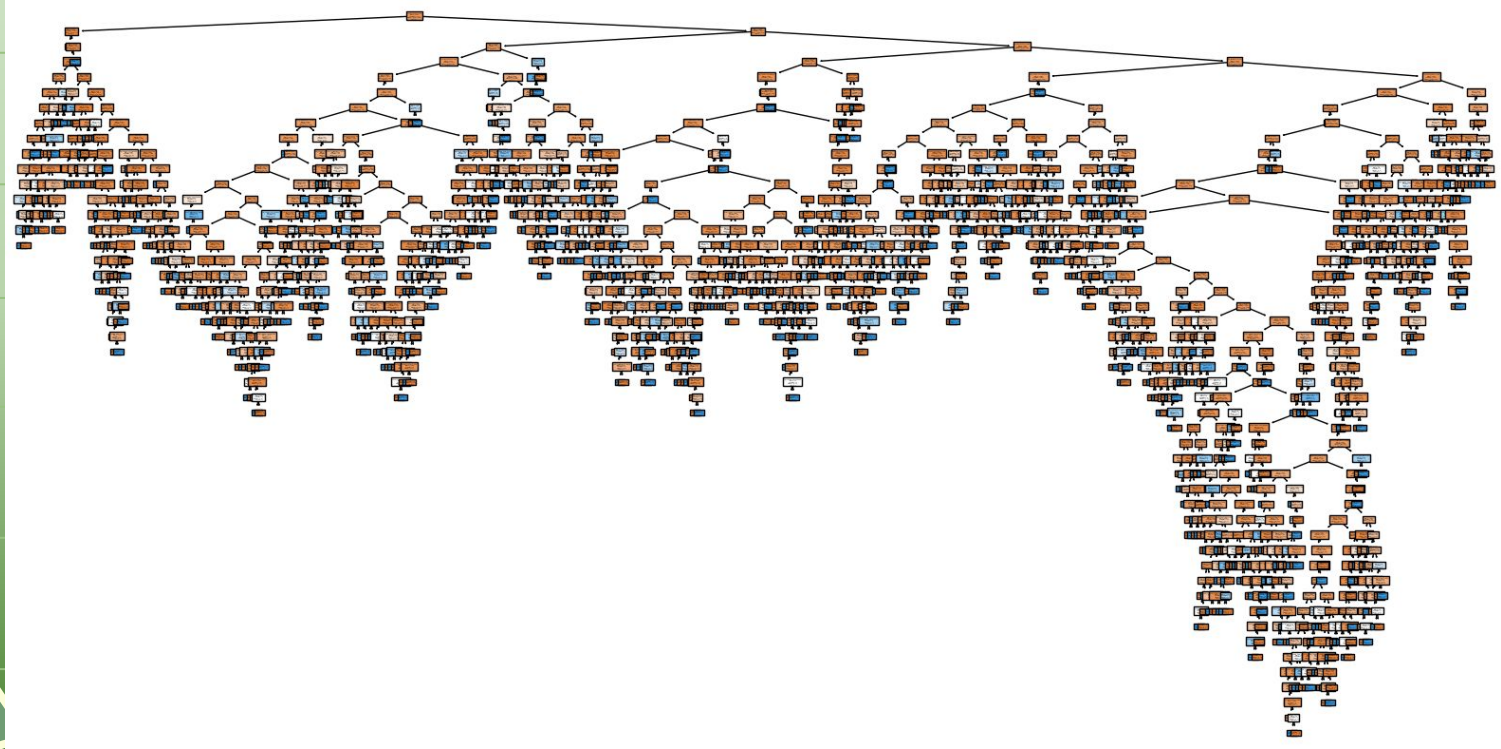
```
[[1269 838]
 [ 175 146]]
```

Test Report:

	precision	recall	f1-score	support
0	0.88	0.60	0.71	2107
1	0.15	0.45	0.22	321
accuracy			0.58	2428
macro avg	0.51	0.53	0.47	2428
weighted avg	0.78	0.58	0.65	2428



Decision Tree



K-Fold Validation

```
# K-Fold Validation
#Implementing cross validation

k = 5
kf = KFold(n_splits=k, random_state=None)
model = LogisticRegression(solver='liblinear')

acc_score = []

for train_index , test_index in kf.split(X):
    X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
    y_train , y_test = y[train_index] , y[test_index]

    model.fit(X_train,y_train)
    pred_values = model.predict(X_test)

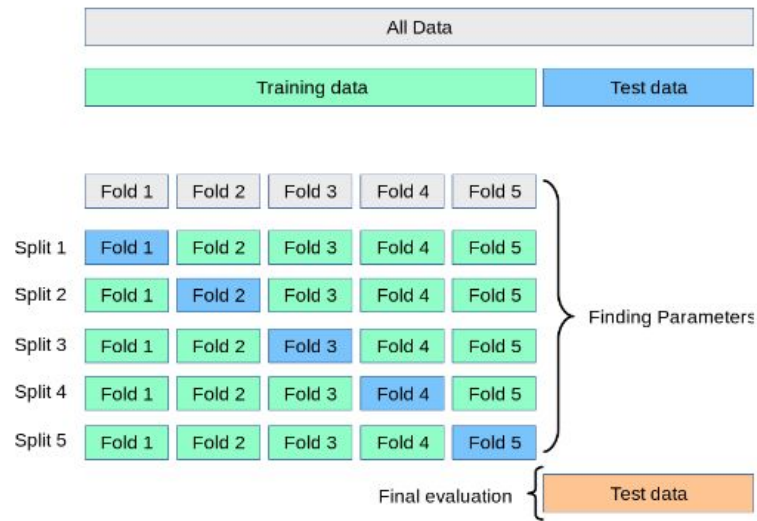
    acc = accuracy_score(pred_values , y_test)
    acc_score.append(acc)

avg_acc_score = sum(acc_score)/k

print('accuracy of each fold - {}'.format(acc_score))
print('Avg accuracy : {}'.format(avg_acc_score))
```

accuracy of each fold - [0.870236869207003, 0.88259526261586, 0.8872296601441813, 0.8789907312049433, 0.8201957753735188]

Avg accuracy : 0.8678496597091012



Machine Learning Model Conclusions

- Main goal: Avoid providing a high risk individual with a loan.
 - Main priority: Minimize the number of false negatives.
- Secondary goal: Limit the number of low risk individuals being denied a loan.
 - Secondary priority: Avoid models with a very high number of false positives.
- Three best models:
 - SVC-Smote with 106 false negatives and 1305 false positives
 - Logistic Regression-Smote with 210 false negatives and 571 false positives
 - KNN-Smote with 175 false negatives and 838 false positives

Visualizations

Housing type

- ☒ Co-op apartment
- ☒ House / apartment

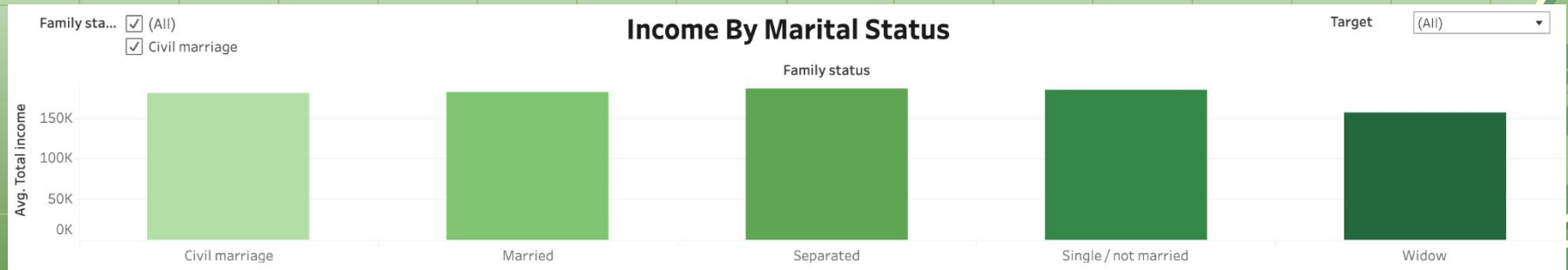
Family Status By Residency Type

Housing type

Family status	Co-op apartment	House / apartment	Municipal apartment	Office apartment	Rented apartment	With parents
Civil marriage	202,500	180,810	159,660	273,000	190,832	162,434
Married	224,591	181,591	183,949	203,742	203,172	170,988
Separated	99,000	186,932	173,905	211,500	183,214	184,500
Single / not married	165,214	184,739	177,750	210,971	196,671	176,418
Widow	135,000	156,981	162,818		94,500	112,500

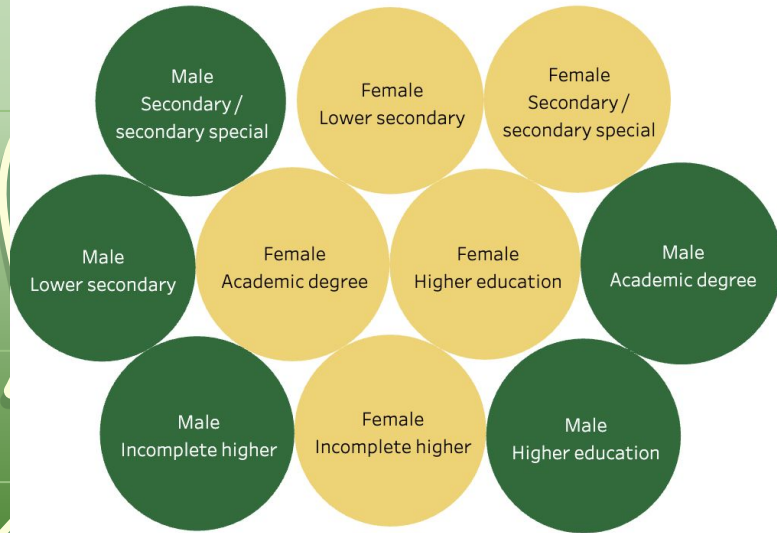
Visualizations cont.

Separated people appear to have the highest income, while those who are widowed have the lowest income.



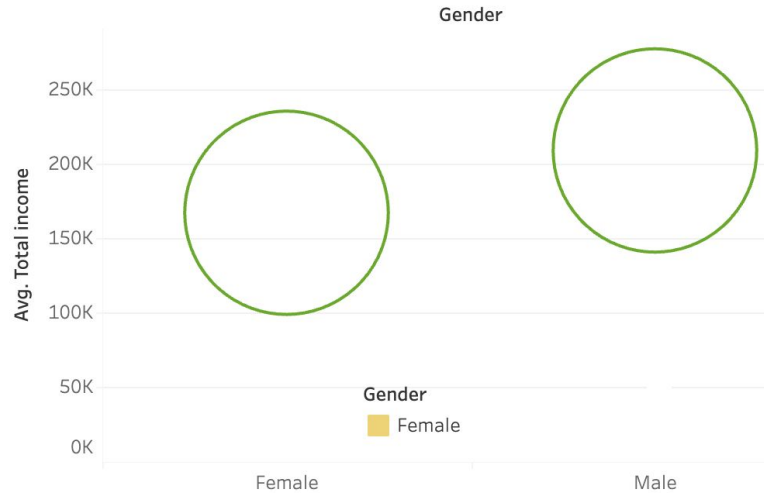
Visualizations cont.

Income By Education



Men, in general, make a higher income than that of their female counterparts.

Income By Gender



Creating the Dashboard

- Our Web App has six pages: a home page, Tableau/visualization page, ML Form, our report, about us, and a works cited page
- On the ML Form we have 15 user inputs (9 dropdowns and 6 fill-ins)

<https://sphilli.pythonanywhere.com/>

Findings

Ultimately, our model is not a great predictor of loan risk based on demographics. Our study highlights the profound impact of class imbalance on the performance of classification models.

Below are a few conclusions we were able to draw based on our visualizations:

Widowed and single applicants seem to be slightly more high-risk than married or separated applicants.

Married and separated applicants appear to be lower-risk.

Separated males appear to be the lowest-risk demographic than other groups.

Men, in general, have a higher income than their female counterparts.

Future Work

- If we had more time we would have liked to continue trying more models, possibly an Ada Boost which may improve our results.
- We would also like to explore other datasets, and compare results to see if demographics are really not a good indicator of loan risk.
- We would also have liked to add a donut chart to the ML Form page showing the user's score in a visualization, as well as eye catching graphics like a green checkmark if they were approved.

Resources

<https://www.kaggle.com/datasets/samuelcortinhas/credit-card-classification-clean-data/data>

<https://www.kaggle.com/datasets/itssuru/loan-data>

https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html

<https://datascience.stackexchange.com/questions/47852/visualizing-decision-tree-with-feature-names>

<https://www.askpython.com/python/examples/k-fold-cross-validation>

<https://www.turintech.ai/what-is-imbalanced-data-and-how-to-handle-it/>