Project Presentation Creditworthiness

SS 24 IDA & ML I – Presented by Kristina Richert

24. September 2024

Overview

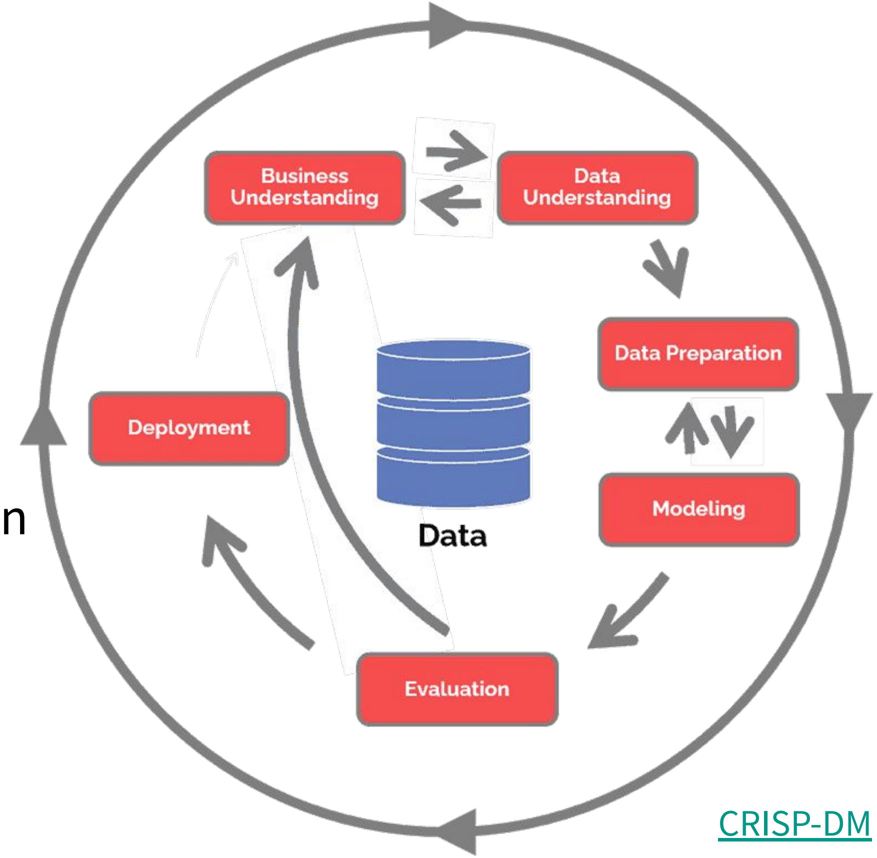
1. Problem Setting

2. Data Exploration
3. Data Preprocessing

4. Modeling:
   4.1. Baseline Training & Evaluation
   4.2. Feature Importance
   4.3. Hyperparameter Tuning

5. Evaluation Result

6. Potential Next Steps



CRISP-DM

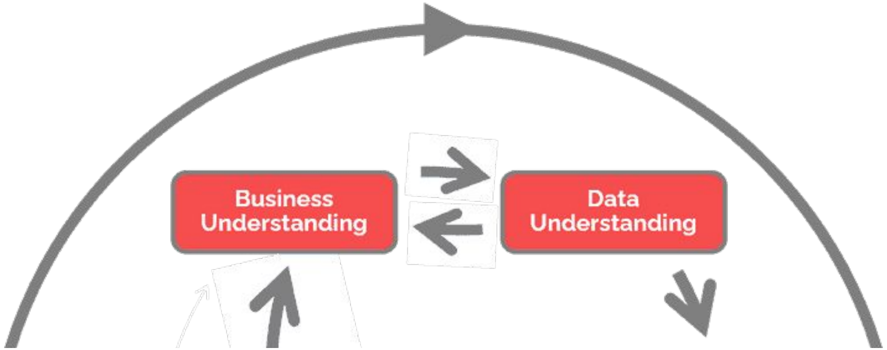
1. Problem Setting

- **Binary Classification:** Predict Creditworthiness of a Bank's customers

- The creditworthiness is known for **1000 customers**

- It is **five times more 'expensive' to wrongfully rate a customer as creditworthy** than vice versa

2. Data Exploration



1. Types of Features and Incompleteness
2. Feature Correlation

2.1 Type of Features and Incompleteness

Creditworthiness - Feature Observations and Preprocessing			
Qualitative / Categorical Data		Quantitative / Numerical Data	
Nominal	Ordinal	Discrete	Continuous
purpose_IN	status_existing_account	installment_rate_disposable_income	duration_month
foreign_worker_IN	credit_history	present_residence_since	credit_amount
personalstatus_sex	savings_account	numb_existing_credit	age_years
other_debtors	current_employment_IN	number_people_being_liable	
property	job_IN		
other_installment_plans			
housing			
telephone			

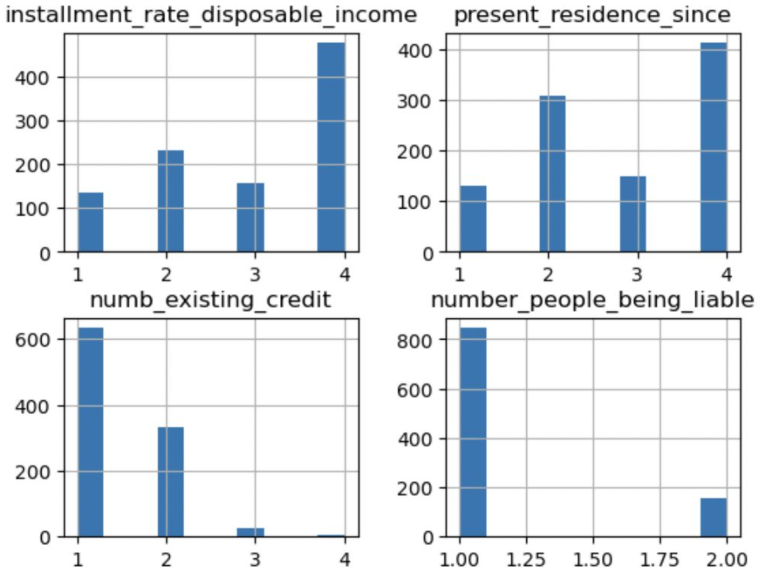
Observations
Nominal, Incomplete
Nominal, Complete
Ordinal, Complete
Ordinal, Incomplete
Continuous, Not Binned, Complete
Discrete, Binned, Complete

- Total 1000 Rows with 21 Features
- Qualitative
  - Nominal Data - 8 Features
  - Ordinal Data - 5 Features
- Quantitative
  - Discrete Data - 4 Features
  - Continuous Data - 3 Features

2.1.1 Quantitative Features

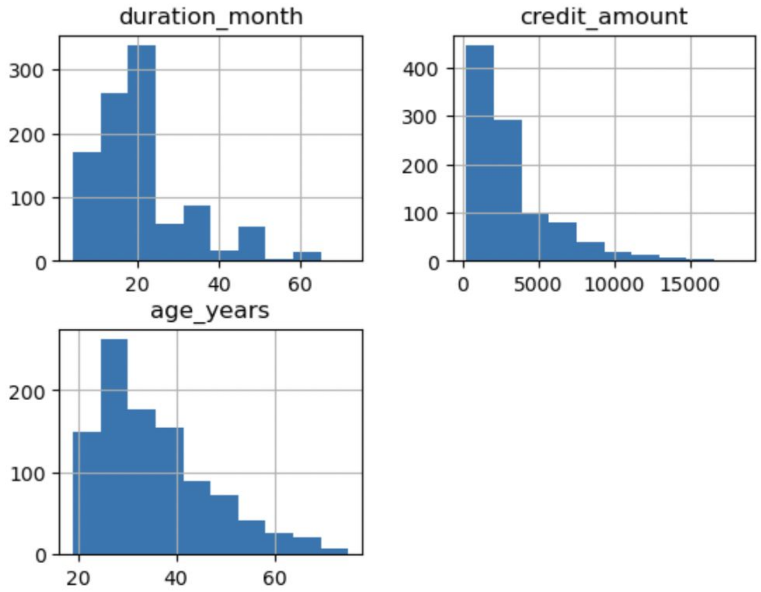
Discrete Features

```
continuous_ft_pre_binned_cols = ['installment_rate_disposable_income',
                                 'present_residence_since',
                                 'numb_existing_credit',
                                 'number_people_being_liable']
semantic_raw_data[continuous_ft_pre_binned_cols].hist();
```



Continuous Features

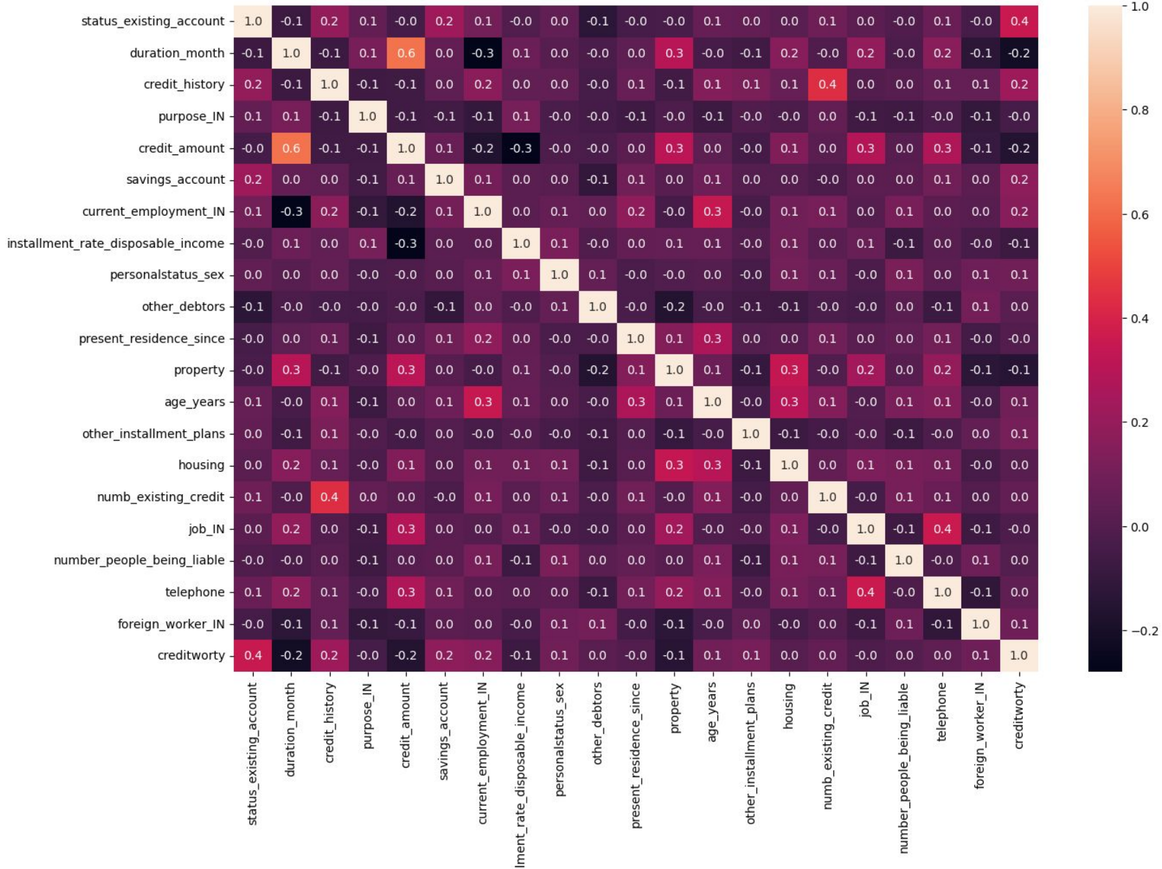
```
continuous_ft_to_bin_cols = ['duration_month',
                             'credit_amount',
                             'age_years']
semantic_raw_data[continuous_ft_to_bin_cols].hist();
```



2.2 Feature Correlation

- **H1:** Is there any strong correlation between incomplete features that could make imputation better?

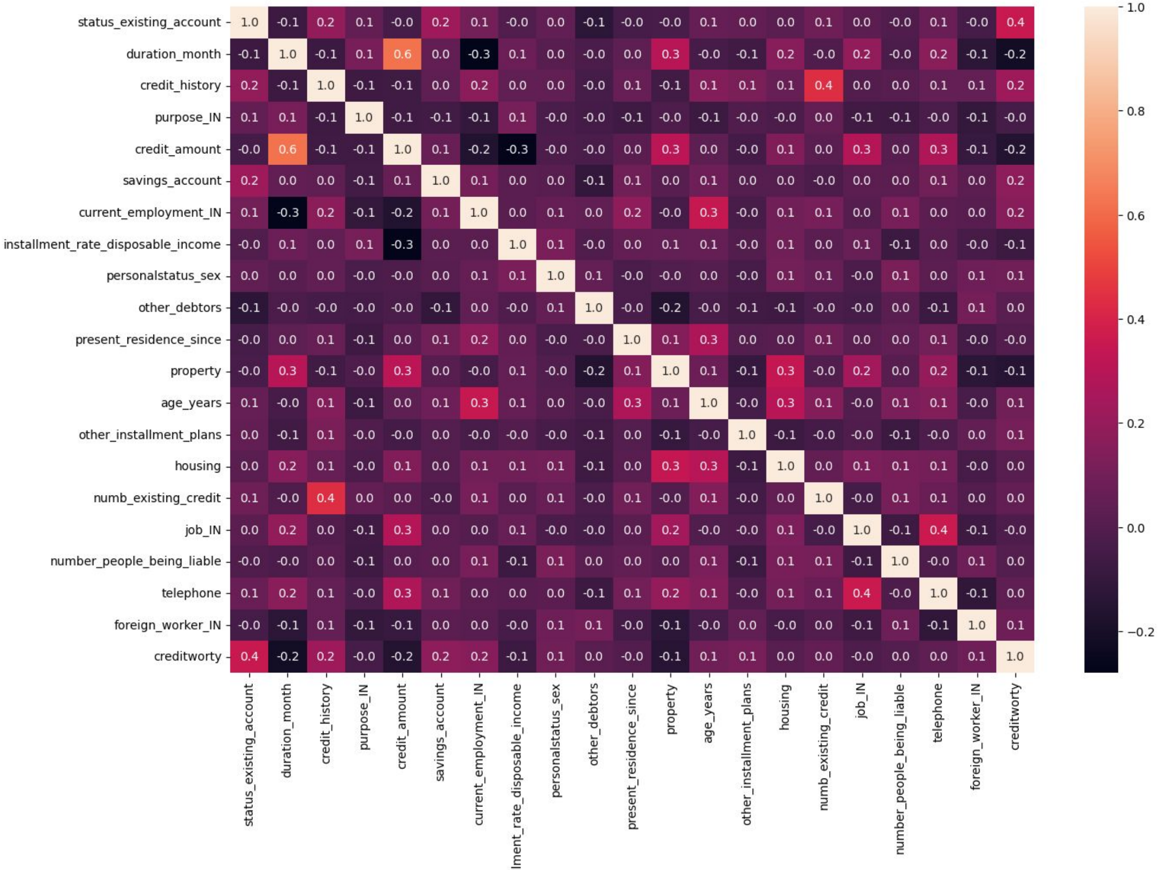
- **H2:** Are there ‘perfectly correlated’ features can be removed?



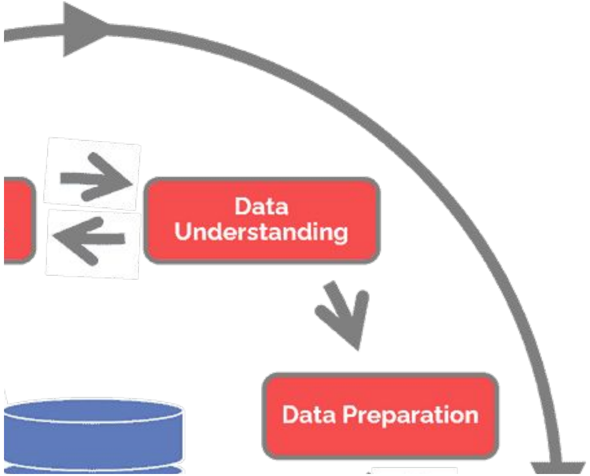
2.2 Feature Correlation

- **H1:** No strong correlation between incomplete features.

- **H2:** No perfectly correlated features.



3. Data Preprocessing



1. Label Encoding
2. Imputing
3. Binning
4. One-Hot Encoding
5. Addressing Class Imbalance

3.1 Label Encoding

- Convert categorical variables into numerical variables
- out of 21 features → 13 are categorical
- applying Label Encoding on these 13 features
- Keeping all the NaN values in the respective elements

Observations	Label Encoding
Nominal, Incomplete	Yes
Nominal, Complete	Yes
Ordinal, Complete	Yes
Ordinal, Incomplete	Yes
Continuous, Not Binned, Complete	No
Discrete, Binned, Complete	No

3.2 Imputing with Linear Regression

- Linear Regression Imputation
  - missing data – dependent variables
  - independent variables used to predict missing data
  - fit model & impute
- Benefits
  - takes relationship between variables into account
  - capture underlying patterns

```
df_origin[incomplete_feautures].head()
```

	purpose_IN	current_employment_IN	job_IN	foreign_worker_IN
0	NaN	NaN	A173	NaN
1	A46	A73	NaN	A201
2	A40	A74	NaN	NaN
3	A43	A73	A173	A201
4	A40	A74	A173	A201

```
imputed_data[incomplete_feautures].head()
```

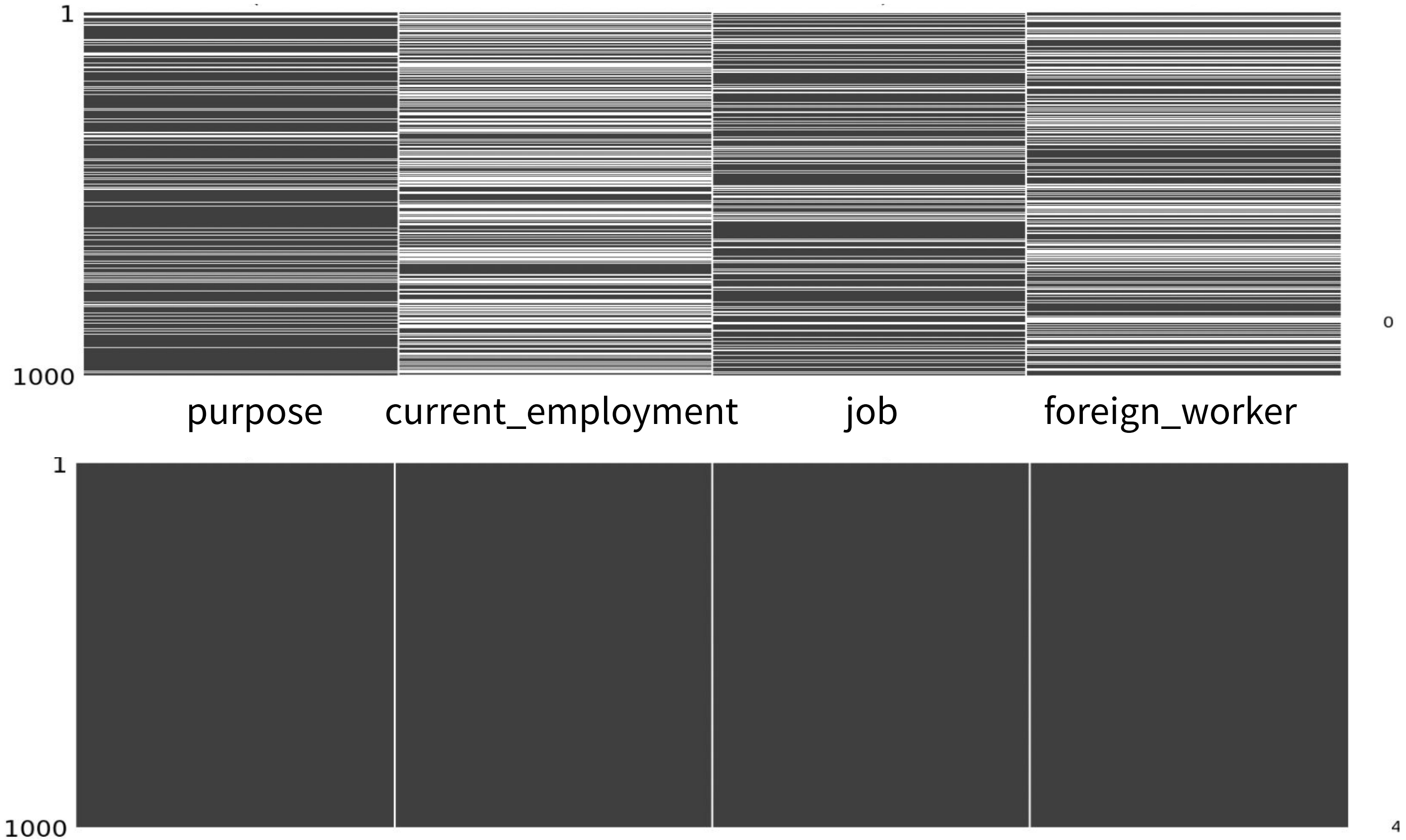
	purpose_IN	current_employment_IN	job_IN	foreign_worker_IN
0	4	3	2	0
1	7	2	2	0
2	0	3	2	0
3	4	2	2	0
4	0	3	2	0

Observations	Imputing with Linear Regression
Nominal, Incomplete	Yes
Nominal, Complete	No
Ordinal, Complete	No
Ordinal, Incomplete	Yes
Continuous, Not Binned, Complete	No
Discrete, Binned, Complete	No

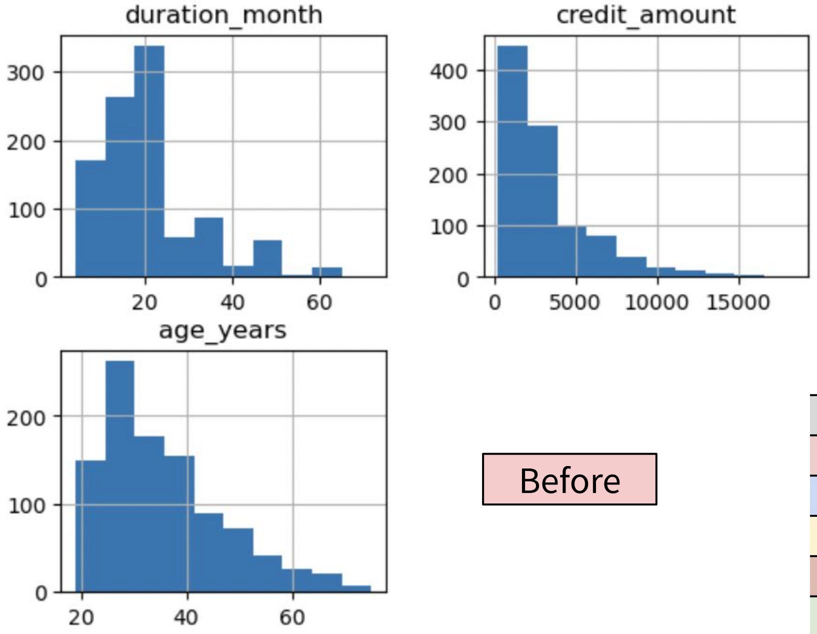
3.2 Imputing with Linear Regression

Before

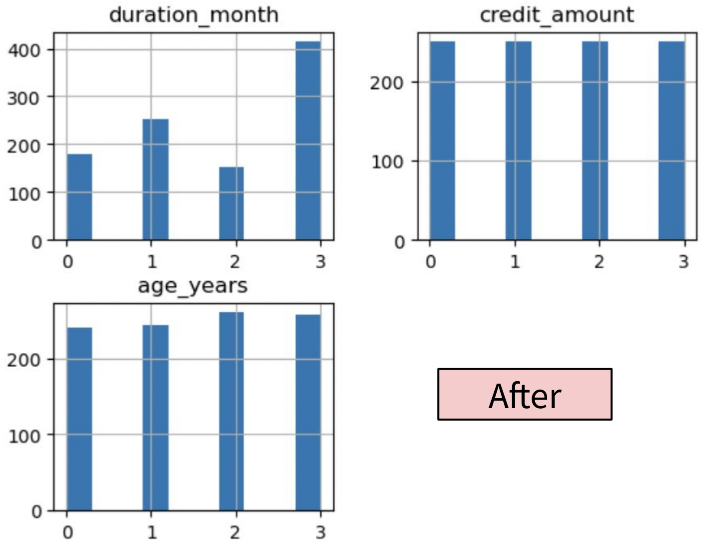
After



3.3 Binning into Buckets



Before



After

Observations	Binning into Buckets
Nominal, Incomplete	No
Nominal, Complete	No
Ordinal, Complete	No
Ordinal, Incomplete	No
Continuous, Not Binned, Complete	Yes
Discrete, Binned, Complete	No

3.4 One-Hot Encoding

- useful for categorical variables without ordinal relationship
- binary variable per unique label
- Apply one-hot encoding on categorical data except for ordinal features

  → from 20 to 63 columns

Observations	One Hot Encoding
Nominal, Incomplete	Yes
Nominal, Complete	Yes
Ordinal, Complete	No
Ordinal, Incomplete	No
Continuous, Not Binned, Complete	Yes
Discrete, Binned, Complete	Yes

Before

property
A123
A124
A124
A121
A123

After

property_0	property_1	property_2	property_3
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0
0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0

Features: Data Preprocessing Summary

Observations	Label Encoding	Imputing with Linear Regression	Binning into Buckets	One Hot Encoding
Nominal, Incomplete	Yes	Yes	No	Yes
Nominal, Complete	Yes	No	No	Yes
Ordinal, Complete	Yes	No	No	No
Ordinal, Incomplete	Yes	Yes	No	No
Continuous, Not Binned, Complete	No	No	Yes	Yes
Discrete, Binned, Complete	No	No	No	Yes

Creditworthiness - Feature Observations and Preprocessing			
Qualitative / Categorical Data		**Quantitative / Numerical Data**	
Nominal	**Ordinal**	**Discrete**	**Continuous**
purpose_IN	status_existing_account	installment_rate_disposable_income	duration_month
foreign_worker_IN	credit_history	present_residence_since	credit_amount
personalstatus_sex	savings_account	numb_existing_credit	age_years
other_debtors	current_employment_IN	number_people_being_liable	
property	job_IN		
other_installment_plans			
housing			
telephone			

3.5 Addressing Class Imbalance

Original Class Balance:

Creditworthy: 700	**Not Creditworthy: 300**

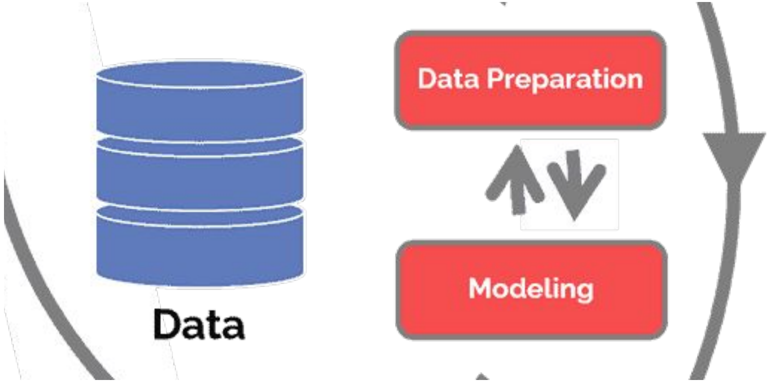
- Imbalanced datasets: biased models that perform poorly on minority classes
- Resampling Techniques:

1. Random Undersampling – reduce from majority class
2. Random Oversampling – increase size of minority class
3. **Synthetic Minority Oversampling Technique (SMOTE)**
    a. data generation through interpolation not duplication (k-neighbor)

Final Class Balance:

Creditworthy: 700	**Not Creditworthy: 700**

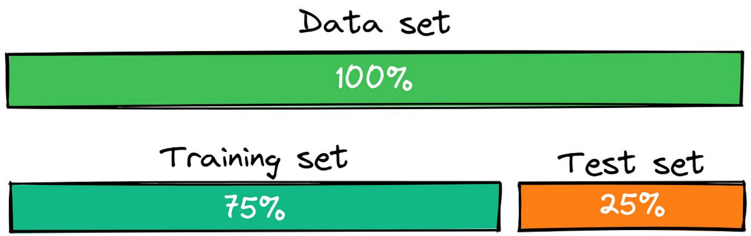
4. Modeling



1. Baseline Training & Evaluation
   a. Splitting and Scaling Training Data
   b. Baseline Training Experiment: Multiple Models
2. Feature Importance
3. Hyperparameter Tuning

4.1.1 Splitting & Scaling Training Data

- Splitting Data into training and test set (75/25)
  - how well does the model generalize to unseen data



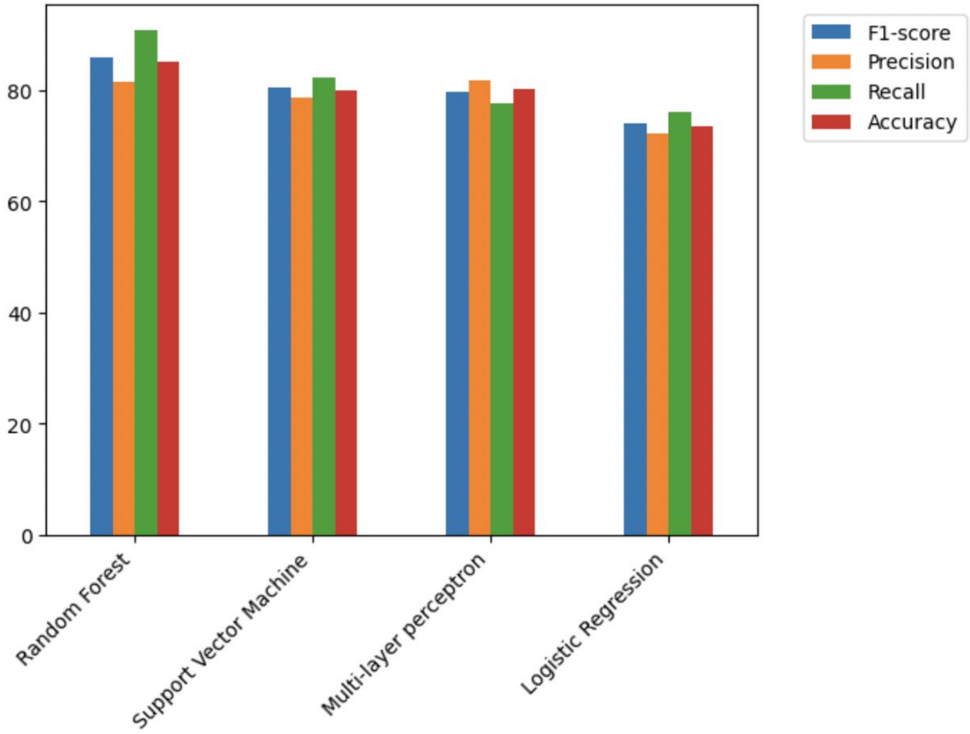
- Scaling the data
  - transform values to specific range
  - ensure all features contribute equally to the result

4.1.2 Baseline Training Experiment: Multiple Models

Model	Benefits	Disadvantages
Logistic Regression	easy to implement, interpret and train	constructs linear boundaries, assumption of linearity between dependent & independent variable
Support Vector Machine	effective in high dimensional space, linear & nonlinear, kernel trick	does not perform well with a lot of noise, not easy to find right parameters
Random Forest	resistant to noise and outliers, manages high-dimensional datasets, high accuracy	computational complex, longer training period
Multi-Layer Perceptron	deal with complex patterns, non-linear activation functions	prone to overfitting when dataset is small, gradient vanishing/exploding

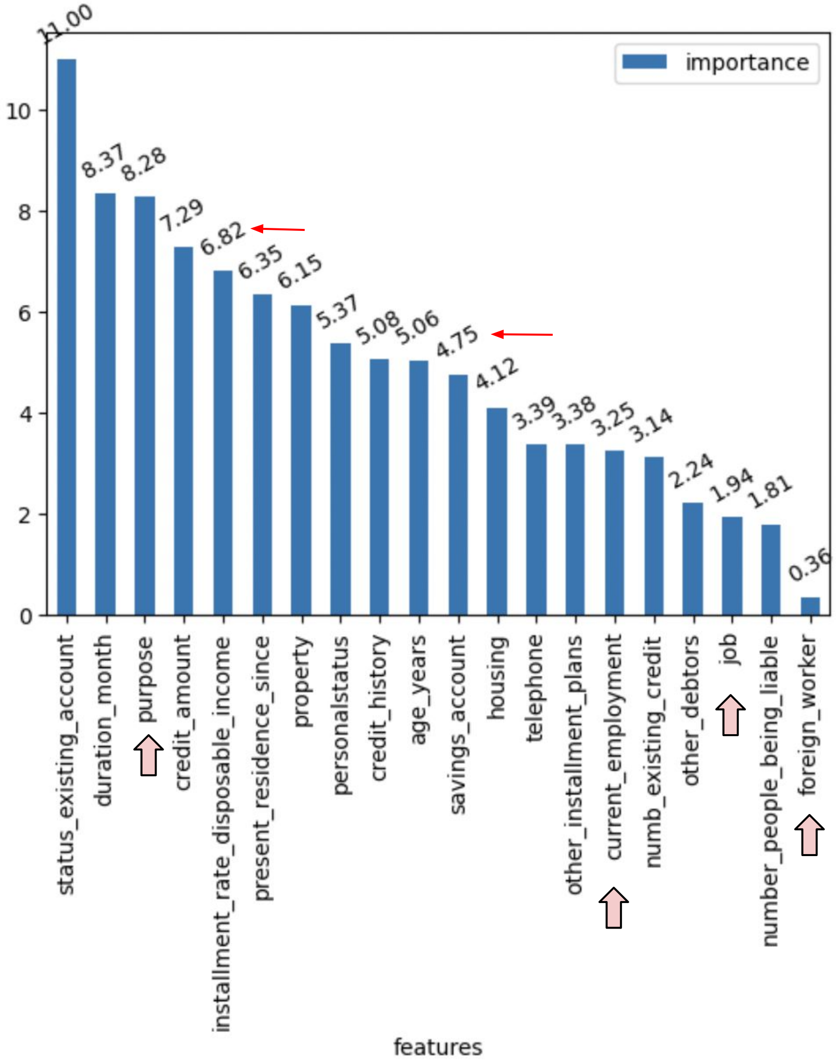
4.1.2 Baseline Training Experiment: Multiple Models

	F1-score	Precision	Recall	Accuracy
Random Forest	85.946	81.538	90.857	85.143
Support Vector Machine	80.447	78.689	82.286	80.000
Multi-layer perceptron	79.765	81.928	77.714	80.286
Logistic Regression	74.095	72.283	76.000	73.429



4.2 Feature Importance

- The feature of **existing status account** has the highest impact on creditworthiness with **11%**.
- The feature of **purpose** also has high importance of 8.2% but unfortunately contains imputed values due to **incompleteness**.
- Will **dimensionality reduction** boost performance in this case?



4.2 Feature Importance

Reducing Features lower than **5%**

→ Dropped **10 Features**

	F1-score	Precision	Recall	Accuracy
Random Forest	87.399	82.323	93.143	86.571
Support Vector Machine	82.955	82.486	83.429	82.857
Multi-layer perceptron	81.657	84.663	78.857	82.286
Logistic Regression	78.161	78.613	77.714	78.286

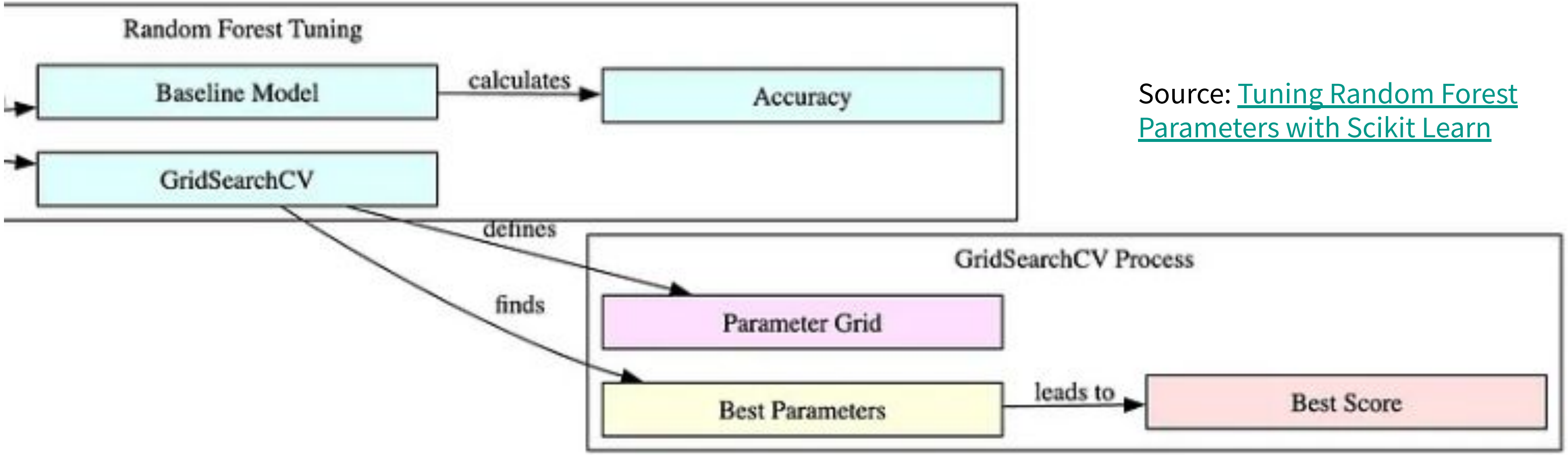
Reducing Features lower than **7%**

→ Dropped **16 Features**

	F1-score	Precision	Recall	Accuracy
Random Forest	73.504	73.295	73.714	73.429
Support Vector Machine	70.552	76.159	65.714	72.571
Logistic Regression	70.030	72.840	67.429	71.143
Multi-layer perceptron	68.308	74.000	63.429	70.571

Due to **lack of significant boost in prediction metrics**, continuing **without** dropping dimensions. Could make sense to drop it if model gets too heavy for deployment.

4.3 Hyperparameter Tuning



Source: [Tuning Random Forest Parameters with Scikit Learn]

```
print('Best hyperparameters are: '+ str(rf_model_random_ini.best_params_))
print('Best score is: '+ str(rf_model_random_ini.best_score_))
```

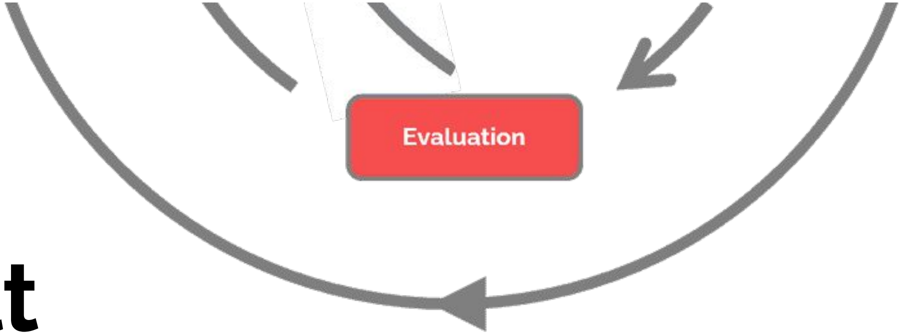
```
Best hyperparameters are: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'log
2', 'max_depth': 40, 'criterion': 'gini', 'bootstrap': False}
Best score is: 0.8558203028136223
```

4.3 Hyperparameter Tuning

```
random_grid = {
    'n_estimators': [int(x) for x in np.linspace(start=100, stop=1000, num=10)],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None] + [int(x) for x in np.linspace(10, 110, num=11)],
    'min_samples_split': [2, 5, 10, 20, 30],
    'min_samples_leaf': [1, 2, 4, 6, 8, 10],
    'bootstrap': [True, False],
    'criterion': ['gini', 'entropy']
}

rf_model_random_ini = RandomizedSearchCV(estimator = RandomForestClassifier(),
                                         param_distributions = random_grid,
                                         n_iter = 100,
                                         cv = 10,
                                         verbose=1,
                                         random_state=42,
                                         n_jobs = -1,
                                         scoring='f1')
rf_model_random_ini.fit(X_train, Y_train)
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
```

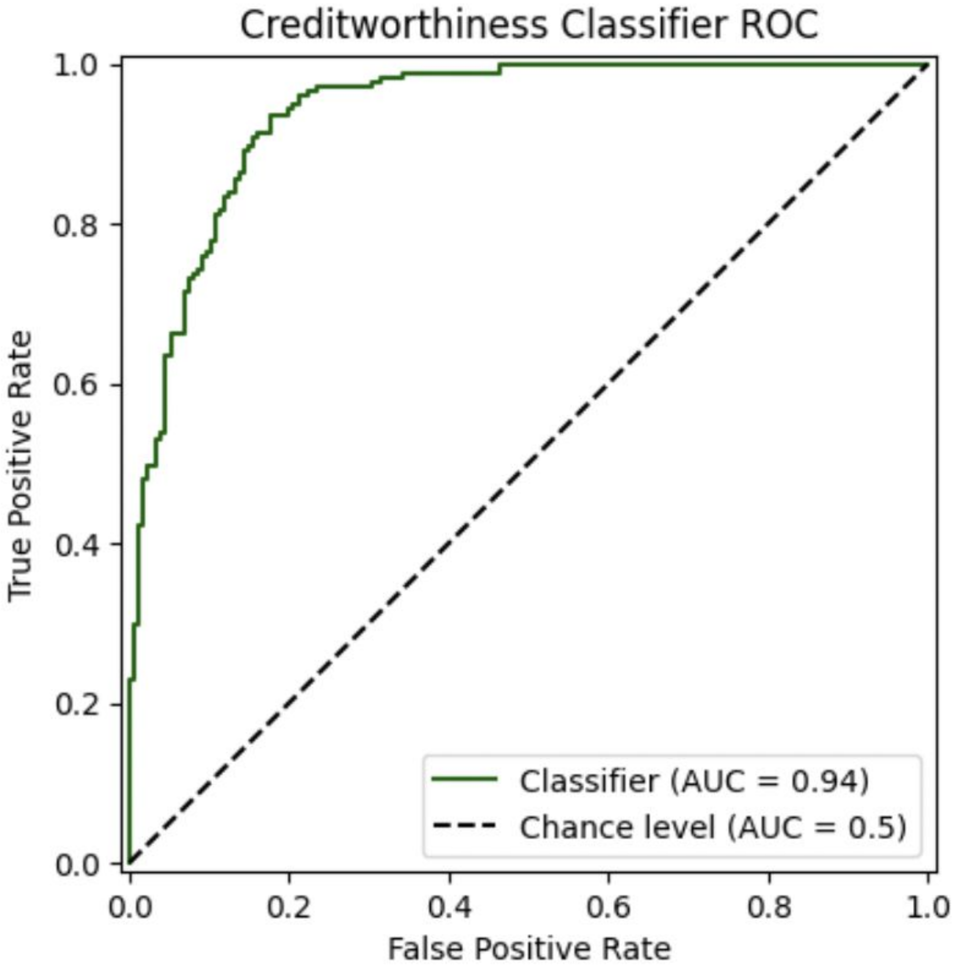


5. Evaluation Result

1. ROC Curve
2. F-Score and Confusion Matrix

5.1 ROC Curve

- Visual Representation of model performance **across all thresholds.**
- Evaluate performance of a decision function in binary classification tasks
- Each Point: threshold value – **trade-off between TP & FP**
- Area under the ROC curve (AUC) Value: **94.047**

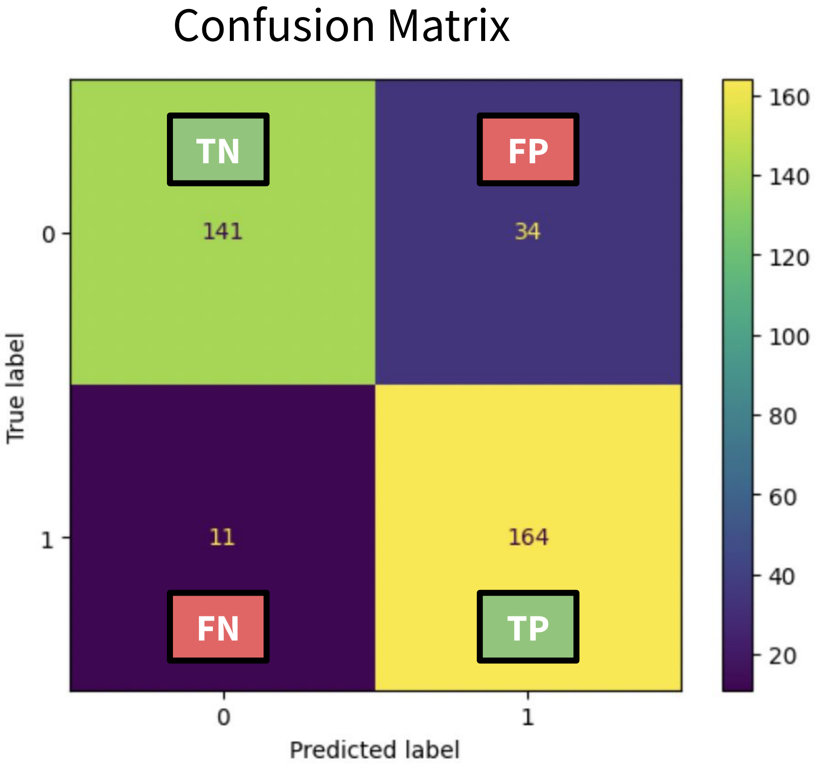


5.1 F-Score and Confusion Matrix

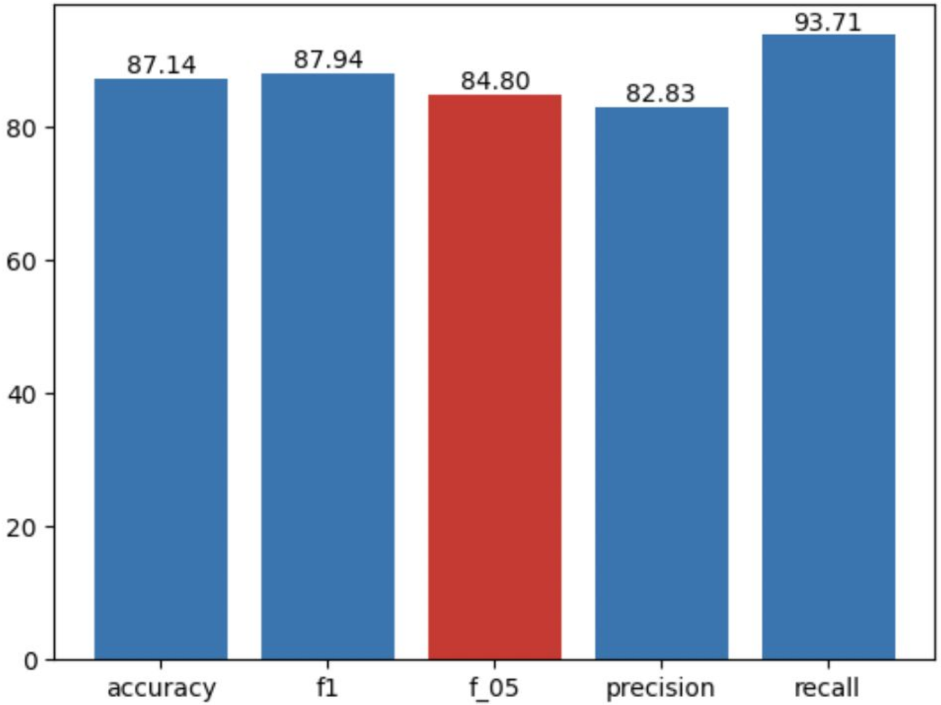
It is five times more 'expensive' to wrongfully rate a customer as creditworthy than vice versa.

What does it mean?

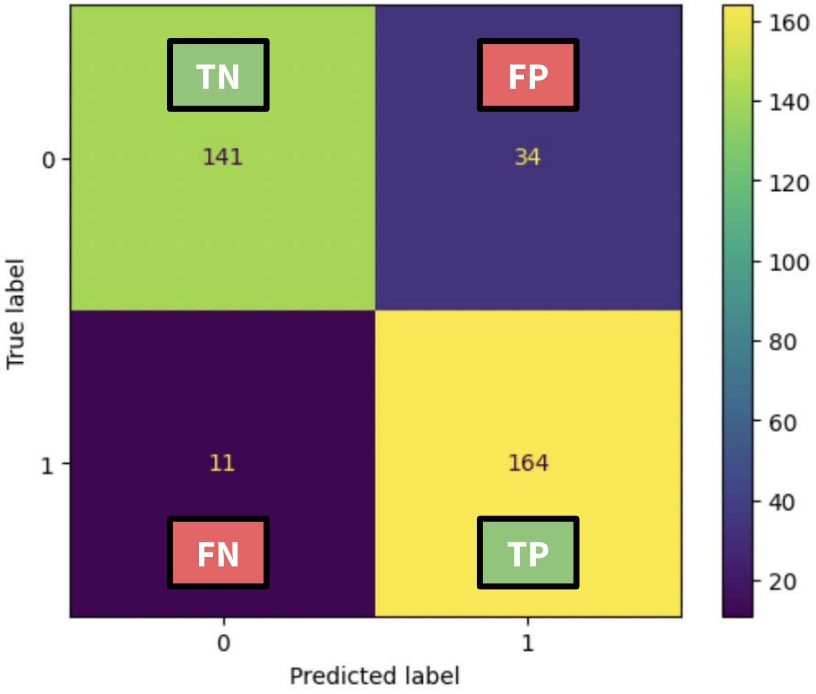
- **5x** more expensive to have **FP** than **FN**
- **Precision** is more important than Recall
- So **F-0.5** is more important than F-1



5.1 F-Score and Confusion Matrix



Confusion Matrix



6. Potential Next Steps

Potential Next Steps

- Focus on Data Collection Measures
  - Collect more data to avoid synthesizing class imbalance
  - Focus on features with high importance

- Preliminary Analysis: Train multiple models
  - Deep dive to further optimize and compare different models

- Short-Term vs. Long-Term Trade-Off
  - Continue focusing on reducing FP without significantly increasing FN

Toolkit Used

Github Repo containing Notebook