

Preliminary Analysis

Feng Wan & Yining Wang

04/18/2025

Introduction

Accurate credit risk assessment is essential for financial institutions seeking to make informed lending decisions. Traditionally, these assessments rely on standardized scoring systems and expert judgment. However, with the growing availability of structured credit data and advances in machine learning, predictive modeling has emerged as a valuable tool for evaluating borrower reliability with greater speed and objectivity.

This project explores how supervised machine learning algorithms can be used to classify loan applicants as either good or bad credit risks. By leveraging historical data that captures key financial and demographic attributes, we aim to build models capable of generalizing to future applicants and supporting risk-aware decision-making in credit systems.

The guiding research question is: **How effectively can individual-level financial and personal attributes be used to predict creditworthiness using classification algorithms?**

To address this, we implement and compare a set of binary classification algorithms along with permutation—including Random Forest, XGBoost, K-Nearest Neighbors, and SVM—evaluating each model's accuracy and interpretability. In addition to assessing model performance, we also investigate the relative importance of various features in predicting credit outcomes, offering insights into which characteristics are most predictive of borrower behavior.

This work contributes to the broader field of applied data science by demonstrating how classification techniques can be used to support socially and economically significant decisions such as consumer credit allocation.

Data

Source and Description

The dataset used in this analysis is the German Credit Data from the UCI Machine Learning Repository, originally compiled by Professor Hans Hofmann. It comprises information on 1,000 individuals, each evaluated for credit risk, and is commonly used as a benchmark for credit scoring and classification tasks in machine learning.

Each case represents a unique loan applicant. The associated variables describe financial and demographic characteristics believed to be relevant to assessing creditworthiness. The target variable classifies each applicant as either a good credit risk (1) or bad credit risk (2).

Data Collection

Key attributes include:

1. **Duration in months of credit:** Indicates the length of the credit agreement a
2. **Credit amount:** The total loan requested by the applicant
3. **Installment rate:** Expressed as a percentage of the applicant's disposable income
4. **Savings account status:** A categorical indicator of the applicant's financial reserves
5. **Employment length:** Reflects job tenure and thus employment stability
6. **Housing type:** Indicates whether the applicant rents
7. **Job category:** Describes the type or status of the applicant's employment
8. **Personal status and sex:** Includes marital status and gender
9. **Purpose of credit:** Indicates the intended use of the loan
10. **Number of existing credits:** Reflects the applicant's current debt load and borrowing behavior
11. **Foreign worker status:** Identifies whether the applicant is a foreign national

Methodology

XGBoost Tree Structure

XGBoost is selected due to its superior performance in structured data tasks, particularly for binary classification problems such as credit risk assessment. XGBoost is a scalable and efficient implementation of gradient boosting that has consistently delivered state-of-the-art results in numerous data science competitions and real-world applications. XGBoost can capture complex relationships. Creditworthiness is rarely determined by a single variable. XGBoost builds an ensemble of decision trees, allowing it to learn non-linear interactions between variables—such as how employment length might affect risk differently depending on credit amount or loan purpose. Also, XGBoost generates feature importance metrics that allow us to rank and interpret the predictive value of each variable, offering actionable insights for financial decision-making.

XGBClassifier

1 ?

(https://xgboost.readthedocs.io/en/release_3.0.0/python/python_api.h

XGBClassifier(alpha=10, base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, feature_weights=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=1.0, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=4, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=100, n_jobs=None)

```
##
## Classification Report:
```

##		precision	recall	f1-score	support
##					
##	0	0.59	0.44	0.51	86
##	1	0.80	0.88	0.84	214
##					
##	accuracy			0.75	300
##	macro avg	0.70	0.66	0.67	300
##	weighted avg	0.74	0.75	0.74	300

From the classification report, we can see that for class 1 (bad credit risk), the model performed well with a precision of 0.80, recall of 0.88, and an F1-score of 0.84. This indicates the model is effective at correctly identifying high-risk individuals. For class 0 (good credit risk), the performance is noticeably weaker, with a recall of 0.44 and an F1-score of 0.51—meaning a considerable number of good applicants were misclassified as risky. This discrepancy likely reflects class imbalance or a model bias that favors predicting the majority class.

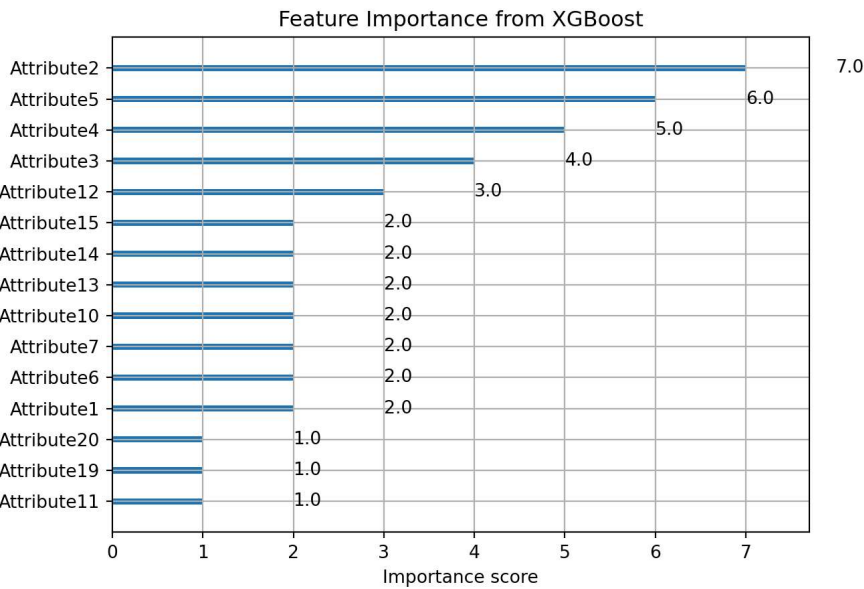
The macro average F1-score is 0.67, and the weighted average is 0.74, suggesting decent overall performance but revealing an uneven ability to classify across both classes accurately.

##	train-auc-mean	train-auc-std	test-auc-mean	test-auc-std
## 0	0.626016	0.005889	0.585627	0.015298
## 1	0.669135	0.028180	0.586173	0.034616
## 2	0.705129	0.024399	0.631407	0.043218
## 3	0.742094	0.014263	0.660823	0.041564
## 4	0.751653	0.010454	0.674828	0.050083
## 5	0.765311	0.017266	0.684732	0.047975
## 6	0.769894	0.016323	0.689245	0.044180
## 7	0.808923	0.017835	0.733557	0.044533
## 8	0.809032	0.016665	0.734908	0.047158
## 9	0.816583	0.017843	0.743292	0.038388
## 10	0.823636	0.018363	0.751078	0.036446
## 11	0.824032	0.017671	0.759482	0.040559
## 12	0.825134	0.018629	0.764732	0.040309
## 13	0.830199	0.019497	0.768162	0.041268
## 14	0.828101	0.018756	0.770302	0.039451
## 15	0.830263	0.018687	0.771789	0.040034
## 16	0.835019	0.018099	0.779356	0.041131
## 17	0.837613	0.018543	0.779749	0.040118
## 18	0.839404	0.018394	0.780710	0.040805
## 19	0.838505	0.017762	0.782773	0.039339
## 20	0.839503	0.017642	0.782549	0.039563
## 21	0.839295	0.016735	0.782305	0.038220
## 22	0.839329	0.016663	0.782161	0.037834
## 23	0.839071	0.016435	0.783553	0.037946

auc score

For the 'train-auc-mean', it started at approximately 0.66 and reached 0.82, indicating strong model learning capacity. The 'test-auc-mean', it reaches a peak value of 0.76, which suggests that the model generalizes well to unseen data and is not severely overfitting. The standard deviation for both the training auc and the testing auc are all tends to be small, suggesting stability in performance across different data splits.

Feature Importance analysis

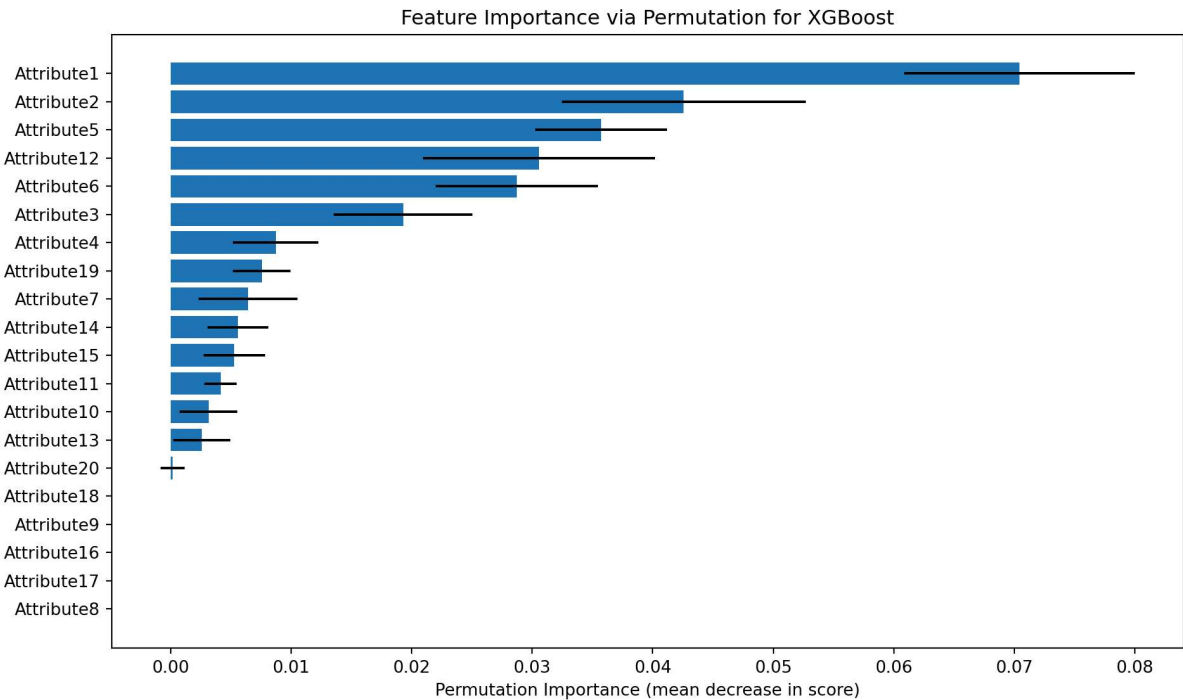


The feature importance plot reveals which variables contributed most frequently to splits in the XGBoost model's decision trees.

For **Attribute2** (Duration in Month), **Attribute5** (Credit Amount), and **Attribute4** (Purpose) scored the highest in terms of importance. These attributes were critical in the early splits of decision trees, indicating their strong predictive power.

For **Attribute20** (Foreign Worker), **Attribute19** (Telephone), and ***Attribute11** (Present Residence Since) has the lowest predicting power.

Permutation



To further validate the importance of each feature in our classification task, we applied permutation importance, a model-agnostic method that measures how much the model's performance metric deteriorates when each feature's values are randomly shuffled. From the bar plot, we can deduce:

For **Attribute1** (Status of Existing Checking Account), **Attribute2** (Duration in Month), and **Attribute5**(Credit Amount) yielded the largest mean drops in performance when permuted, indicating they are the most critical features for predicting credit risk in this model.

These results are broadly consistent with the earlier feature importance rankings from XGBoost's internal metric, supporting their robustness.

Random Forest

We use Random Forest as a baseline model due to its strong performance on classification tasks involving structured, tabular data. As an ensemble of decision trees, Random Forest reduces overfitting by averaging multiple trees trained on random subsets of data and features. This makes it particularly robust to noise and well-suited for our credit dataset, which includes both categorical and numerical variables. Additionally, Random Forest provides feature importance metrics and supports permutation-based interpretability, which aligns with our goal of understanding which applicant attributes are most predictive of credit risk.

Random Forest Accuracy

RandomForestClassifier

i ?

(<https://scikit-learn.org/1.6/modules/generated/sklearn.ensemble.RandomForestClassifier.html>)

RandomForestClassifier(random_state=42)

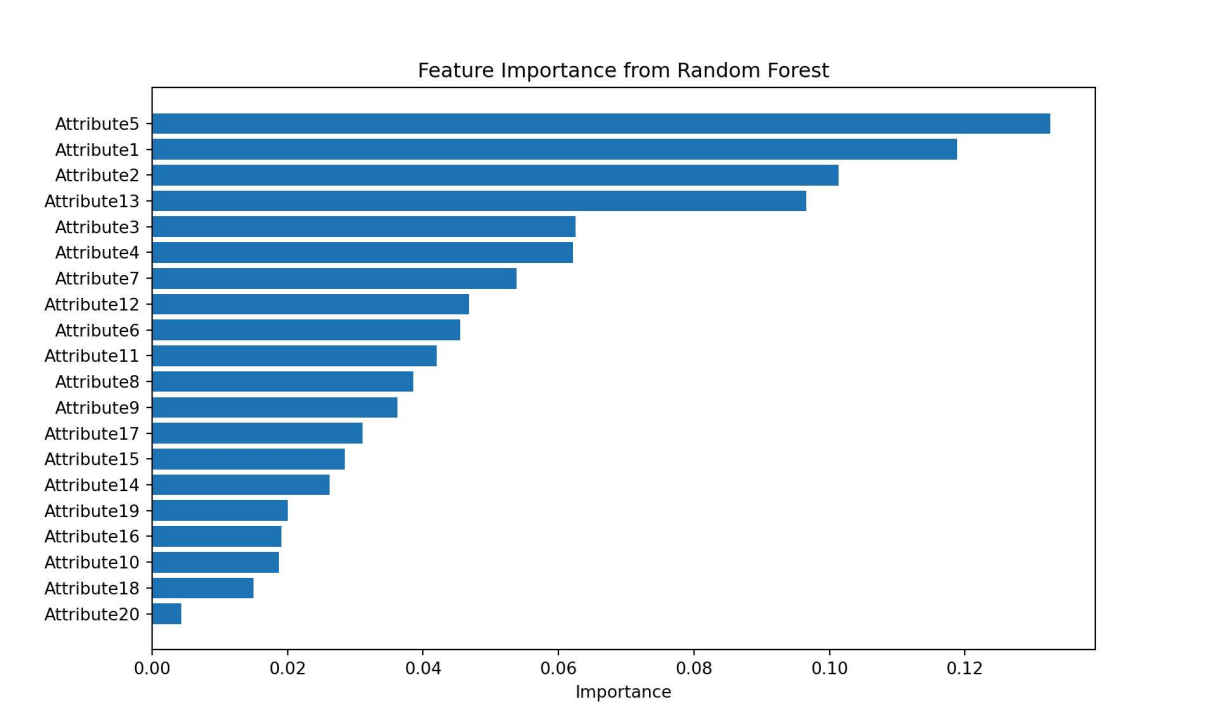
Classification Report:

##		precision	recall	f1-score	support
##					
##	0	0.65	0.48	0.55	86
##	1	0.81	0.90	0.85	214
##					
##	accuracy			0.78	300
##	macro avg	0.73	0.69	0.70	300
##	weighted avg	0.76	0.78	0.77	300

From the classification report, we can see that for class 1 (bad credit risk), the model performed strongly, with a precision of 0.81, recall of 0.90, and an F1-score of 0.85. This suggests the model is highly effective at identifying high-risk individuals. For class 0 (good credit risk), the performance was more modest, with recall dropping to 0.48 and an F1-score of 0.55—indicating that many good applicants were mistakenly classified as risky. This may be due to class imbalance or a tendency of the model to favor the majority class.

The macro-average F1-score is 0.70 and the weighted-average F1-score is 0.77, reflecting decent overall performance but still showing room for improvement in detecting the minority class.

Feature Importance from Random Forest



The feature importance plot generated from the trained Random Forest model ranks each attribute based on its relative contribution to reducing impurity (Gini importance) across all trees in the ensemble.

For **Attribute5** (Credit Amount), **Attribute1** (Status of Existing Checking Account), and **Attribute2** (Duration in Months) stands out to be the most important, contributing the most to the model's performance. These features likely hold significant information for distinguishing between good and bad credit risks.

For **Attribute10** (Other Debtors / Guarantors), **Attribute18** (Number of People Being Liable to Provide Maintenance for), and **Attribute20** (Foreign Worker) have the lowest importance, contributing the least.

Feature Importance via Permutation for Random Forest

Attribute	Permutation Importance (mean decrease in score)
Attribute1	0.122
Attribute2	0.044
Attribute5	0.036
Attribute6	0.031
Attribute13	0.024
Attribute3	0.023
Attribute4	0.022
Attribute12	0.014
Attribute7	0.013
Attribute14	0.011
Attribute18	0.007
Attribute17	0.006
Attribute8	0.006
Attribute11	0.006
Attribute15	0.004
Attribute16	0.003
Attribute9	0.003
Attribute19	0.002
Attribute10	0.002
Attribute20	0.002

Other features had smaller impacts, and some like **Attribute20** (Foreign Worker) had little to no effect, suggesting they may be less useful in this model. The black lines represent standard deviation across permutations and give a sense of consistency in the results.

The K-Nearest-Neighbors has a comparison model because it offers a simple, intuitive, and non-parametric approach to classification. KNN makes predictions based on the most similar data points in the training set, without making strong assumptions about the underlying data distribution. This allows us to explore how well instance-based learning performs on our credit risk dataset, where classification may depend on combinations of multiple features.

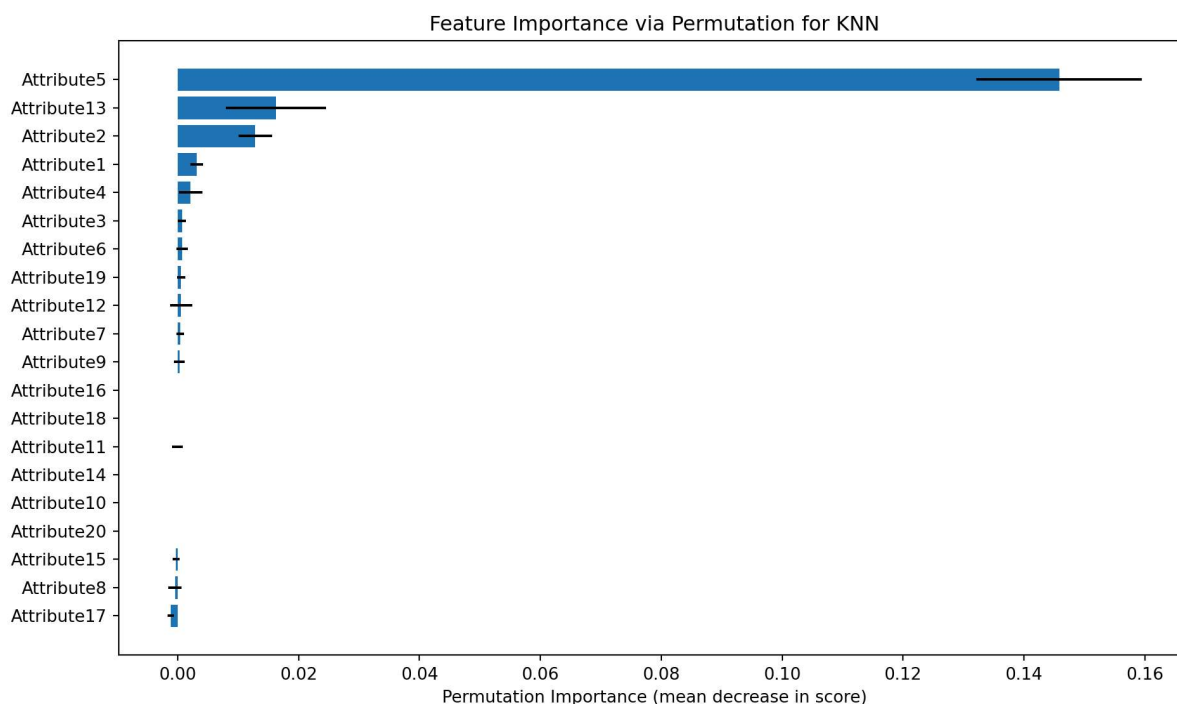
```
▼ KNeighborsClassifier (https://scikit-learn.org/1.6/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)
KNeighborsClassifier()

##
## Classification Report:

##           precision    recall  f1-score   support
##
##      0           0.40      0.27      0.32         86
##      1           0.74      0.84      0.79        214
##
## accuracy                0.68         300
## macro avg              0.57         0.55         300
## weighted avg           0.64         0.68         300
```

The macro-average F1-score is 0.55 and the weighted-average is 0.65, pointing to a moderate overall performance but with a clear skew in how the model treats each class.

Feature Importance with Premutation





For the **Attribute5** (Credit Amount), it is by far the most important feature, showing the highest mean drop in accuracy when permuted. This suggests that KNN relies heavily on this feature when computing distances between instances.

Most other features—including **Attribute11** (Present Residence Since), **Attribute10** (Other Debtors / Guarantors), and **Attribute14** (Other Installment Plans)—have near-zero importance, indicating that they had little to no impact on the model's predictions.

SVM

The SVM is a powerful and widely used algorithm for binary classification tasks, especially effective in high-dimensional feature spaces. SVM works by finding the optimal hyperplane that maximizes the margin between the two classes—in this case, good and bad credit risks.

SVM Accuracy Score and Classification Report

▼ SVC   [\(https://scikit-learn.org/1.6/modules/generated/sklearn.svm.SVC.html\)](https://scikit-learn.org/1.6/modules/generated/sklearn.svm.SVC.html)

SVC()

```
##
## Classification Report:
```

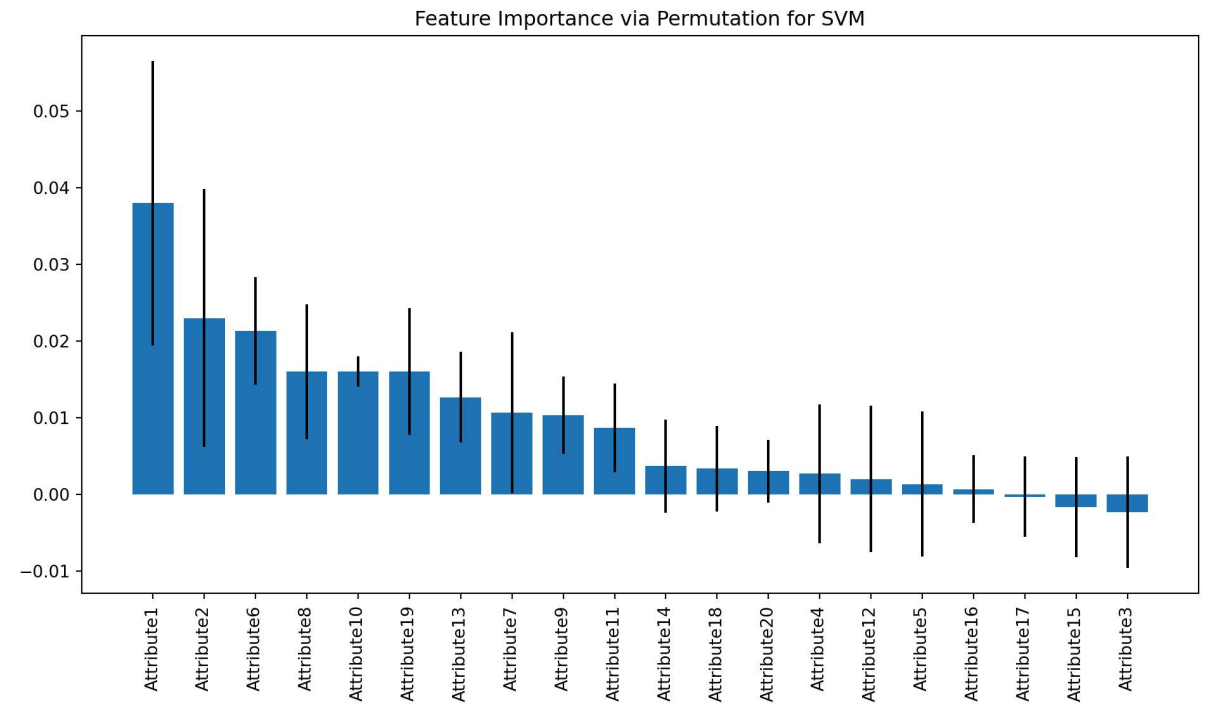
##		precision	recall	f1-score	support
##					
##	0	0.63	0.48	0.54	86
##	1	0.81	0.89	0.85	214
##					
##	accuracy			0.77	300
##	macro avg	0.72	0.68	0.69	300
##	weighted avg	0.76	0.77	0.76	300

SVM achieved an accuracy of 0.77, meaning it correctly classified 77% of the test data. The AUC score is also 0.77, indicating that the model has strong discriminative power in separating good and bad credit risk cases.

From the classification report, we can see that for **class 1** (bad credit risk), the model performed well with a precision of 0.81, recall of 0.89, and an F1-score of 0.85. This suggests the model is effective at identifying high-risk individuals. For **class 0** (good credit risk), performance was weaker, with recall dropping to 0.48, meaning more false negatives (good applicants misclassified as risky). This could reflect class imbalance or a margin-maximizing bias toward the majority class. The **macro average F1-score** is 0.69, and the weighted average is 0.76, showing decent overall performance but highlighting some imbalance between the two classes.

Feature Importance with Permutation

```
## ([<matplotlib.axis.XTick object at 0x00000231F80A5F10>, <matplotlib.axis.XTick object at 0x00000231F0EA7B60>, <matplotlib.axis.XTick object at 0x00000231F0E8FF50>, <matplotlib.axis.XTick object at 0x00000231F822EA50>, <matplotlib.axis.XTick object at 0x00000231F0794920>, <matplotlib.axis.XTick object at 0x00000231F822DB80>, <matplotlib.axis.XTick object at 0x00000231F822F8F0>, <matplotlib.axis.XTick object at 0x00000231F822F9E0>, <matplotlib.axis.XTick object at 0x00000231F8248AA0>, <matplotlib.axis.XTick object at 0x00000231F822FF80>, <matplotlib.axis.XTick object at 0x00000231F8248D70>, <matplotlib.axis.XTick object at 0x00000231F8249970>, <matplotlib.axis.XTick object at 0x00000231F824A0F0>, <matplotlib.axis.XTick object at 0x00000231F824A990>, <matplotlib.axis.XTick object at 0x00000231F0E63FB0>, <matplotlib.axis.XTick object at 0x00000231F8249CD0>, <matplotlib.axis.XTick object at 0x00000231F824BAA0>, <matplotlib.axis.XTick object at 0x00000231F8268170>, <matplotlib.axis.XTick object at 0x00000231F824BE00>], [Text(0, 0, 'Attribute1'), Text(1, 0, 'Attribute2'), Text(2, 0, 'Attribute6'), Text(3, 0, 'Attribute8'), Text(4, 0, 'Attribute10'), Text(5, 0, 'Attribute19'), Text(6, 0, 'Attribute13'), Text(7, 0, 'Attribute7'), Text(8, 0, 'Attribute9'), Text(9, 0, 'Attribute11'), Text(10, 0, 'Attribute14'), Text(11, 0, 'Attribute18'), Text(12, 0, 'Attribute20'), Text(13, 0, 'Attribute4'), Text(14, 0, 'Attribute12'), Text(15, 0, 'Attribute5'), Text(16, 0, 'Attribute16'), Text(17, 0, 'Attribute17'), Text(18, 0, 'Attribute15'), Text(19, 0, 'Attribute3')])
```



Attribute1 (Status of Existing Checking Account) stands out as the most important feature, with a noticeable mean decrease in score and relatively wide confidence bounds. Other top contributors include **Attribute2** (Duration in Months), **Attribute6** (Savings Account/Bonds), and **Attribute8** (Installment Rate), which all caused moderate drops in performance when permuted.

Result

To address our research question—how accurately can we classify credit risk using financial and demographic features—we applied four machine learning models: XGBoost, Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).

Each model revealed insights into which features were most informative:

1. **XGBoost** highlighted **Attribute2**, **Attribute5**, and **Attribute4** as top contributors.
2. **Random Forest** (standard and permutation) consistently ranked **Attribute1**, **Attribute2**, and **Attribute5** as most important.
3. **KNN** relied heavily on **Attribute5**, with little contribution from other variables—highlighting its sensitivity to local distance and feature scaling.
4. **SVM** showed **Attribute1**, **Attribute2**, and **Attribute6** as having the strongest influence on classification outcomes.

Interpretation

XGBoost and **Random Forest** were the most balanced models in terms of accuracy and interpretability.

SVM offered high recall for detecting bad credit risk, making it valuable for applications where minimizing false negatives is critical.

KNN, while conceptually simple, underperformed and demonstrated reliance on a single key feature, making it less robust for this task.

Discussion

Our goal was to classify individuals as good or bad credit risks using machine learning. Among the models we tested, **Random Forest** and **SVM** performed best, each achieving about 77% accuracy. **SVM** also had strong recall for detecting high-risk individuals. Feature importance analysis consistently highlighted **Attribute1**, **Attribute2**, and **Attribute5** as key predictors.

While the models performed well on the dataset, there are limitations. The data is small, from a single country and time period, which restricts generalizability. Without an external dataset, we cannot guarantee that the model will perform similarly on new data.

Ethically, credit scoring models must be used carefully. Poorly designed models can amplify existing biases or unfairly penalize individuals. However, if properly validated, such models can support fairer and faster lending decisions.

Moreover, we acknowledge that there are more binary classification models that could be used like logistic regression. However, we are not applying logistic regression because it assumes a linear decision boundary, which may be too simplistic for this credit classification task where the relationship between features and class labels (good vs. bad credit risk) is likely nonlinear and complex.

In future work, we suggest testing on larger and more recent datasets, applying fairness checks, and improving interpretability through tools like SHAP. Moreover, another improvement is to do feature processing, dropping attributes with small permutation importance and examine that would actually improve the model performance. However, we decided not to do that because the error bars for most of them are beyond 0 boundary, which means that these features are not definitely negatively impacting the model performance. However, we acknowledge that that's certainly something worth trying.