May 2025

Staffordshire University

Machine Learning

PROJECT REPORT

CREDIT CARD FRAUD DETECTION

ABDESAMAD QOUQI
HASSANE RAMDJEE
LILOU CONSTANTIN
LUCIE MOREAU

SUMMARY

01

INTRODUCTION

Introduction & Objective

In today's digital financial ecosystem, the security of payment systems is paramount. Financial institutions face growing threats from fraudulent transactions, which not only cause monetary losses but also damage customer trust and institutional reputation.

Our Machine Learning project was undertaken as a response to a simulated request from a bank (that we called the Staff's Bank) that required a robust, accurate, and scalable machine learning model capable of detecting fraudulent credit card transactions.

Our ultimate goal is to assist such organizations in proactively identifying anomalies and fraudulent behavior from transactional data in real time, enabling swift mitigation.

Our approach involves the development and comparative analysis of several supervised machine learning models (via Google Collab). These models are trained to detect anomalies (in this case fraudulent activity) from a highly imbalanced dataset.

The following report outlines each phase of our project, from data exploration and preprocessing to model training, evaluation, and finally, our insights and recommendations.

Our Team

Hassane Ramdjee

I focused on exploring the dataset and uncovering key insights through data visualizations and statistics. I analyzed patterns, spotted imbalances, and made sure the raw data was clearly understood before modeling.

Abdesamad Qouqi

I took charge of building and tuning the machine learning models. I explored different algorithms, fine-tuned parameters, and ran evaluations to find the best-performing approach for fraud detection.

Lucie Moreau

I worked on preprocessing the data and preparing it for modeling. I handled tasks like scaling, feature transformation, and used SMOTE to balance our dataset, ensuring the models would train effectively.

Lilou Constantin

I focused on evaluating the results and telling the story behind the numbers. I created visual comparisons like ROC curves, interpreted metrics, and wrote insights and recommendations based on model performance.

02

EXPLORATION

Data exploration & Preprocessing

For this project, we used a public dataset from Kaggle called "Credit Card Fraud Detection." It contains real transaction data from European cardholders, recorded over two days in 2013.

The dataset includes 284,807 transactions, each with 30 anonymized features (named V1 to V28), and two extra columns: Time, Amount, and Class. The "Class" column tells whether a transaction is legit (0) or fraudulent (1).

We chose this dataset because it reflects a real-world problem. It's also very challenging because fraud cases make up less than 0.4% of all transactions, which makes it hard for models to learn to detect them.

To make testing faster and still keep the class imbalance, we created a smaller version of the dataset. This new subset (called df_small) includes all fraud cases and a random sample of 9500 legit ones, for a total of 9,992 rows. This keeps things manageable while keeping the data realistic.

When we loaded the data, we saw that there were no missing values, and all features were numerical. That meant we didn't need to worry about missing data or encoding text. We checked some basic statistics, and noticed that some columns had very different ranges. For example, 'Amount' and 'Time' were on completely different scales compared to the anonymized features.

To understand the relationships between features, we created a correlation heatmap. Most of the features didn't show strong relationships with each other, but a few, like V14 and V17, were negatively correlated with the fraud label. These might be helpful for the models.

We also plotted a count chart to show how unbalanced the classes are. Even in the smaller dataset, legit transactions still greatly outnumber frauds. This imbalance could make models favor the majority class, so we needed to fix that.
Before training our models, we standardized all the numerical features using a scaler.

This step helps models like K-Nearest Neighbors and Logistic Regression work better.

Then, during training, we used SMOTE, a method that creates synthetic fraud examples to balance the data. This gave the models a better chance to learn what fraud looks like.

```
=== THE PROJECT IS STARTING... ===


=== Loading Dataset ===

=== Dataset Overview ===
Dataset shape: (9992, 31)

Data types:
 Time      float64
V1         float64
V2         float64
V3         float64
V4         float64
V5         float64
V6         float64
V7         float64
V8         float64
V9         float64
V10        float64
V11        float64
V12        float64
V13        float64
V14        float64
V15        float64
V16        float64
V17        float64
V18        float64
V19        float64
V20        float64
V21        float64
V22        float64
V23        float64
V24        float64
V25        float64
V26        float64
V27        float64
V28        float64
Amount     float64
Class        int64
dtype: object
```

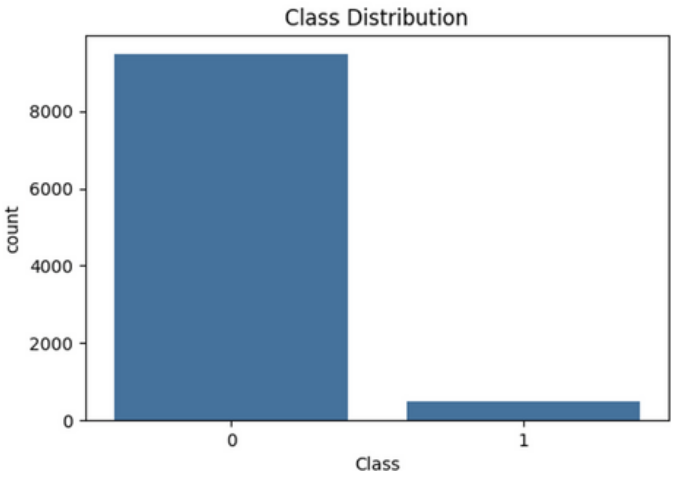
The dataset has 9,992 rows and 31 numerical columns. Features V1 to V28 are anonymized, likely from PCA. The Class column indicates if a transaction is fraudulent (1) or not (0). There is no categorical data so preprocessing is simpler.

```
Missing values:
 Time      0
V1         0
V2         0
V3         0
V4         0
V5         0
V6         0
V7         0
V8         0
V9         0
V10        0
V11        0
V12        0
V13        0
V14        0
V15        0
V16        0
V17        0
V18        0
V19        0
V20        0
V21        0
V22        0
V23        0
V24        0
V25        0
V26        0
V27        0
V28        0
Amount     0
Class      0
dtype: int64
```

No missing values are found in any column. This saves us from using imputation methods and allows direct preprocessing.

```
Class Distribution:
 Class
0    0.950761
1    0.049239
Name: proportion, dtype: float64
```

Around 95% of transactions are normal and only 5% are fraud cases. The class imbalance is severe and requires balancing techniques like SMOTE.

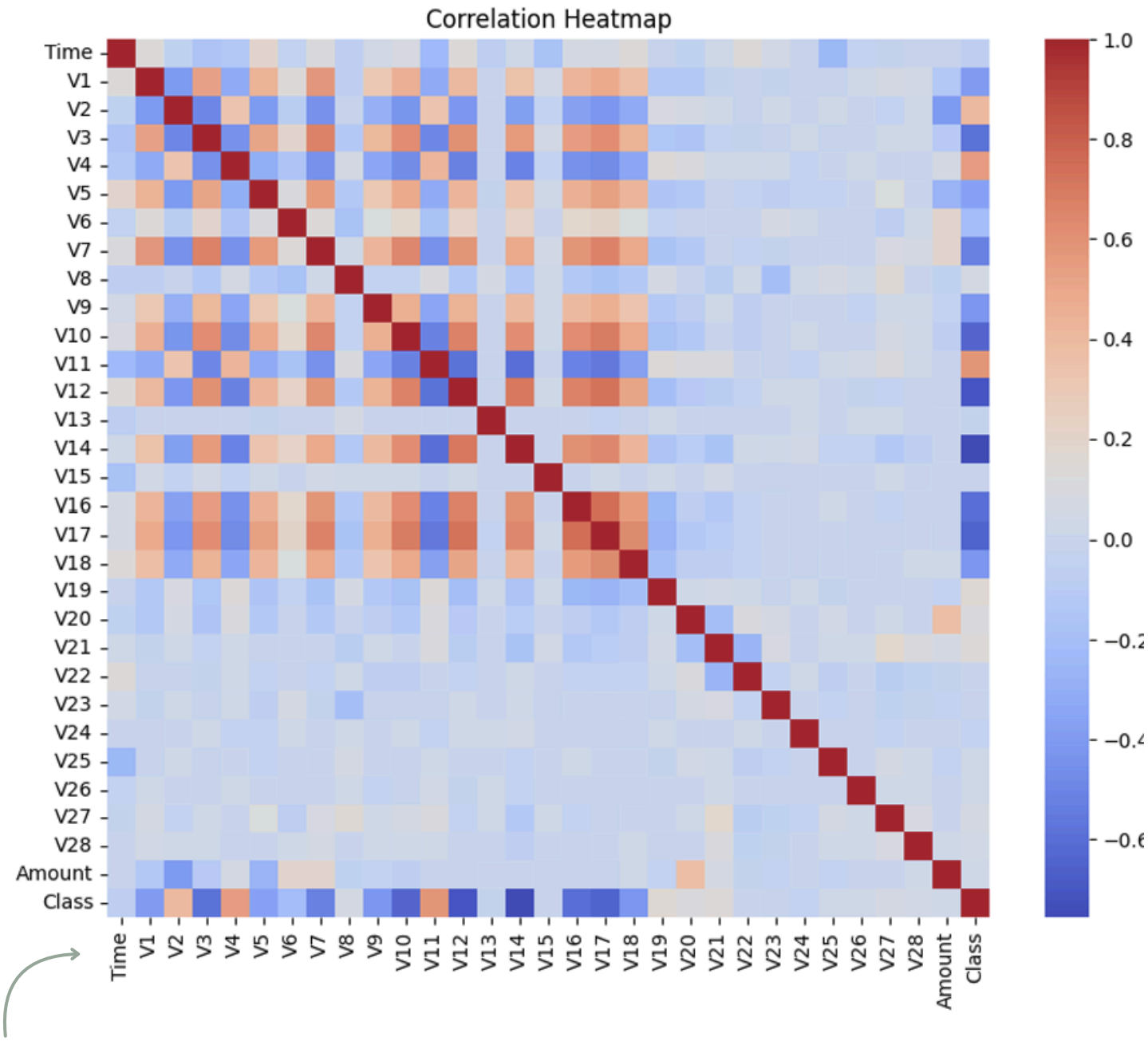


This bar chart highlights the large gap between fraud and non-fraud cases. It helps us justify the use of oversampling to train fair models.

```
Class Distribution:
 Class
0    0.950761
1    0.049239
Name: proportion, dtype: float64

Summary statistics:
```

	count	mean	std	min	25%	50%	75%	max
Time	9992.0	94527.261009	47475.908352	0.000000	53988.000000	85013.000000	138991.500000	172768.000000
V1	9992.0	-0.233419	2.592427	-30.552380	-1.010147	-0.061401	1.281505	2.398119
V2	9992.0	0.180153	1.970990	-29.336007	-0.567822	0.111848	0.897442	22.057729
V3	9992.0	-0.356994	2.600060	-31.103685	-1.074893	0.082126	0.966890	3.770236
V4	9992.0	0.213081	1.802583	-4.790224	-0.806092	0.055539	0.901161	12.114672
V5	9992.0	-0.124424	1.895114	-22.105532	-0.724259	-0.056199	0.637219	28.516513
V6	9992.0	-0.070464	1.400971	-17.282140	-0.820519	-0.300881	0.374729	15.323769
V7	9992.0	-0.261797	2.285010	-43.557242	-0.624710	0.020657	0.576989	15.915767
V8	9992.0	0.034718	1.849676	-41.044261	-0.209994	0.030297	0.356862	20.007208
V9	9992.0	-0.111741	1.333487	-13.434066	-0.713535	-0.076334	0.577086	7.929051
V10	9992.0	-0.268950	1.937235	-24.588262	-0.621730	-0.124901	0.417030	11.768124
V11	9992.0	0.186569	1.405835	-3.423222	-0.722618	0.041590	0.848843	12.018913
V12	9992.0	-0.287096	1.935694	-18.683715	-0.510200	0.101288	0.604003	3.960856
V13	9992.0	-0.004571	0.994782	-3.642446	-0.653882	-0.014922	0.647895	3.530343
V14	9992.0	-0.343748	1.996714	-19.214325	-0.523526	0.016508	0.475734	5.389864
V15	9992.0	-0.013045	0.933515	-4.498945	-0.607804	0.028944	0.649475	3.881395
V16	9992.0	-0.208581	1.482082	-14.129855	-0.555331	0.015363	0.501637	3.801851
V17	9992.0	-0.302488	2.240044	-25.162799	-0.527210	-0.078534	0.405259	8.538195
V18	9992.0	-0.104796	1.143903	-9.498746	-0.553615	-0.021365	0.487429	4.243841
V19	9992.0	0.046403	0.869208	-4.366767	-0.445056	0.027898	0.516496	5.228342
V20	9992.0	0.021572	0.758095	-18.757060	-0.214023	-0.058752	0.153748	13.119819
V21	9992.0	0.034671	1.113052	-22.797604	-0.220868	-0.018758	0.204776	27.202839
V22	9992.0	0.003596	0.771903	-8.887017	-0.539726	0.008730	0.526245	8.361985
V23	9992.0	-0.000693	0.637412	-19.254328	-0.169007	-0.012693	0.153820	13.750136
V24	9992.0	-0.008871	0.609856	-2.836627	-0.361444	0.035048	0.429212	3.951679
V25	9992.0	-0.005373	0.537700	-4.781606	-0.328765	0.017471	0.353634	2.782860
V26	9992.0	0.001634	0.482132	-1.535092	-0.330408	-0.048976	0.255604	2.859167
V27	9992.0	0.009845	0.484792	-7.263482	-0.069848	0.004804	0.106253	4.610936
V28	9992.0	0.004498	0.335822	-8.307955	-0.052267	0.012732	0.087054	15.373170
Amount	9992.0	88.370621	224.700537	0.000000	4.927500	21.000000	79.000000	5627.060000
Class	9992.0	0.049239	0.216378	0.000000	0.000000	0.000000	0.000000	1.000000

Here we have summary stats for every feature, including mean, standard deviation, min/max, and quartiles. Some features have wide ranges or very high standard deviations. Also, the Amount and Time columns are on totally different scales compared to the rest. This tells us we'll need to apply scaling before training our models.



Our heatmap shows how features correlate with each other. Most values are close to zero, meaning low correlation overall. However, some pairs like V2 and V5, or V14 and V17, show moderate to strong correlation. It helps spot redundancy or strong links between features. Also, we see that the target Class has little direct correlation with most features (which is expected in fraud detection problems).

03

MODELING

Model implementation & Evaluation

The "Class" column in our dataset shows whether a transaction is fraudulent or not. Since it has only two possible values (0 or 1), this is a binary classification problem.

To solve it, we trained and tested four different machine learning models:

- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors (KNN)

Each model was chosen for a reason. Logistic Regression is a simple, interpretable model, good for setting a baseline. Decision Trees can handle complex, non-linear patterns. Random Forest, which is a group of decision trees, is usually more powerful and stable. KNN uses distances between data points and works well when features are scaled properly.

To improve performance, we tuned the hyperparameters for each model using RandomizedSearchCV. We used 3-fold stratified cross-validation, which means the data was split into three parts, and each part had the same proportion of fraud and legit transactions. This gave us a fair and balanced way to measure model performance.

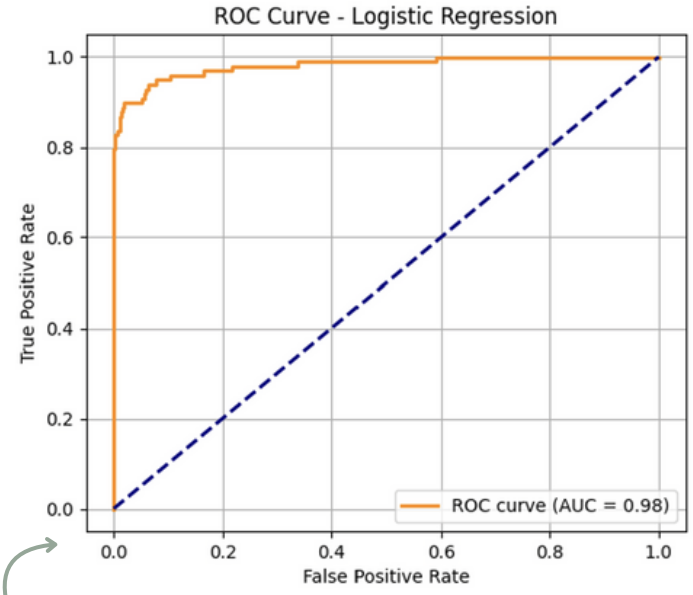
We mainly used the F1-score as our evaluation metric, because it balances precision (how many predicted frauds are actually frauds) and recall (how many actual frauds were caught). In fraud detection, both are important—missing fraud is bad, but too many false alarms are also a problem.

After training, we tested all models on 20% of the data (a hold-out set) that wasn't used during training. We looked at their precision, recall, F1-score, and AUC-ROC (a metric that shows how well a model separates the classes).

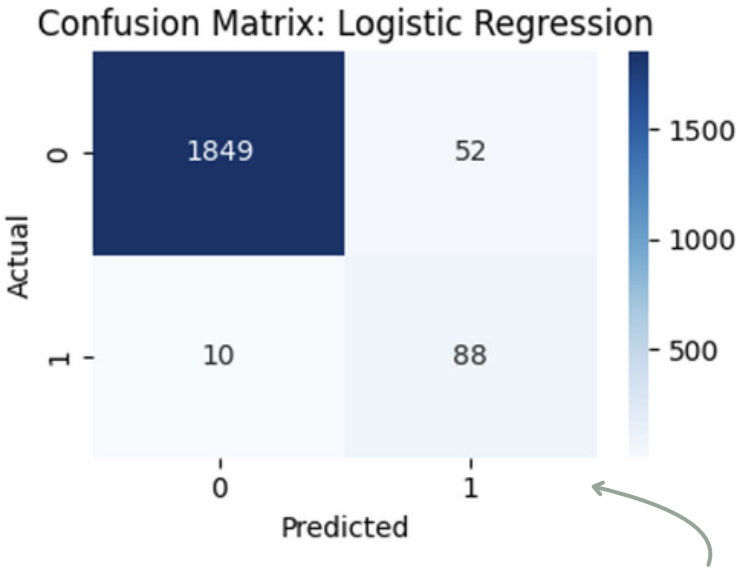
The results showed that Random Forest had the best overall performance. It got the highest F1-score and almost perfect AUC-ROC, meaning it was excellent at telling fraud from legit transactions.
Logistic Regression had the best AUC but struggled with precision. KNN did well on recall but had more false positives. Decision Tree worked okay, but was clearly outperformed by Random Forest.

We also created confusion matrices and ROC curves for all models. The confusion matrices showed that Random Forest had the fewest false positives and false negatives. The ROC curve comparison confirmed that it gave the best trade-off between catching fraud and avoiding mistakes.

Logistic Regression



The ROC curve shows strong model performance, with an AUC of 0.98. This means the logistic regression is very good at separating fraud and non-fraud cases. The curve stays close to the top-left corner, which indicates high true positive rates and low false positives (a great sign for classification).

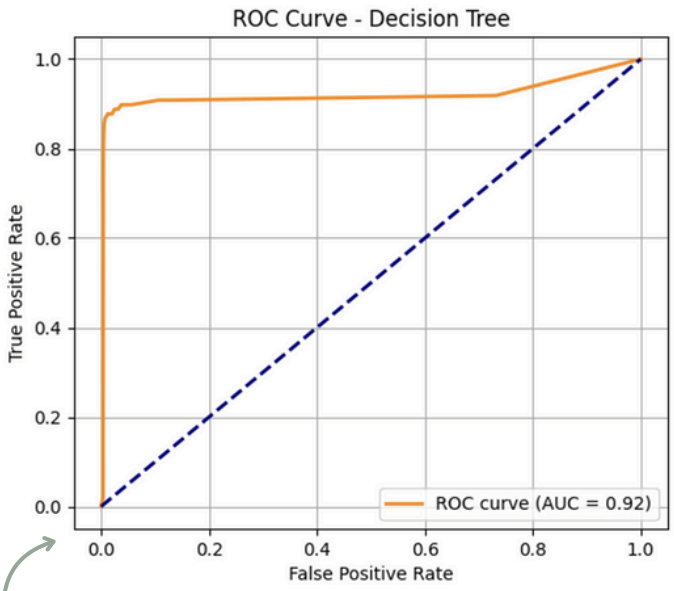


Out of 98 fraud cases, the model correctly identified 88 and missed 10. It also misclassified 52 normal transactions as fraud. The model achieves good recall (few false negatives), which is important in fraud detection (it's better to catch more frauds even if it means a few false alerts).

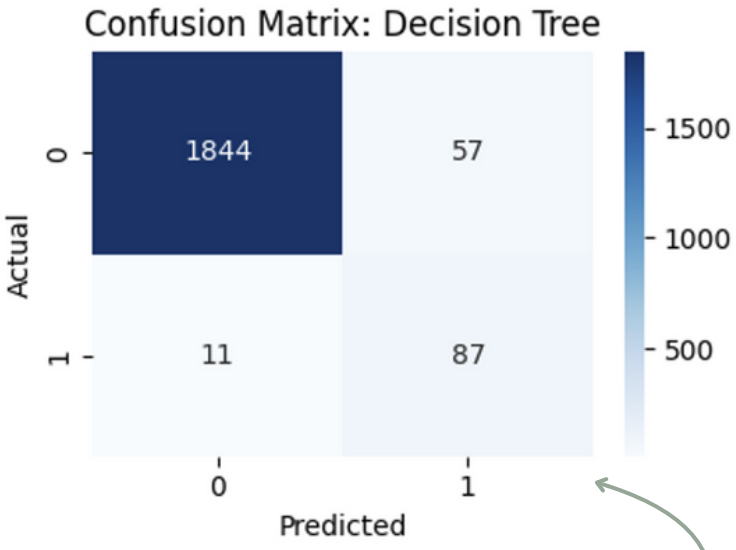
```
Training Logistic Regression...
Rationale: A robust, interpretable baseline for binary classification.
Best Params: {'model__solver': 'liblinear', 'model__C': 0.01}
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1901
           1       0.63      0.90      0.74        98

    accuracy                           0.97      1999
   macro avg       0.81      0.94      0.86      1999
weighted avg       0.98      0.97      0.97      1999
```

Precision for fraud is 0.63, recall is 0.90, and F1-score is 0.74. The model is more sensitive than precise, meaning it captures most frauds but with some false positives. Logistic regression, though simple, performs very well with a 97% overall accuracy, and works especially well in imbalanced scenarios when recall is prioritized.

Decision Tree



The ROC curve shows solid performance, with an AUC of 0.92. The model separates fraud and non-fraud cases well, though with slightly more overlap than logistic regression. The curve's shape indicates high recall and decent precision.

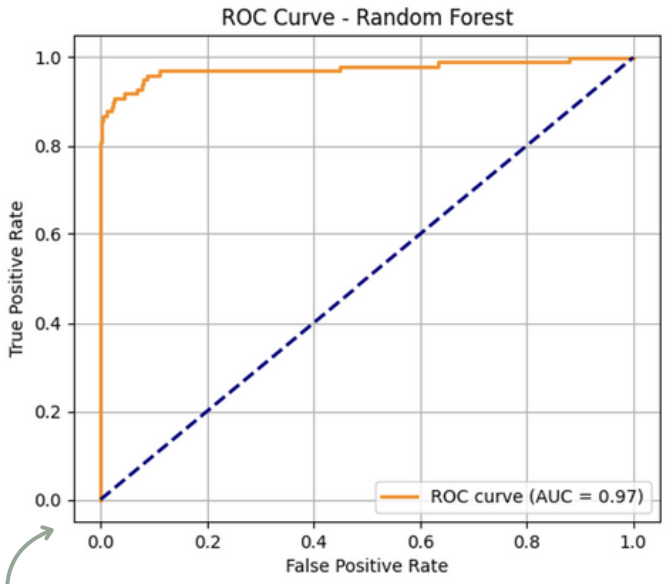


Out of 98 frauds, the model correctly flagged 87 and missed 11. It also incorrectly labeled 57 normal transactions as fraud. While not perfect, it maintains a good balance between catching fraud and limiting false positives.

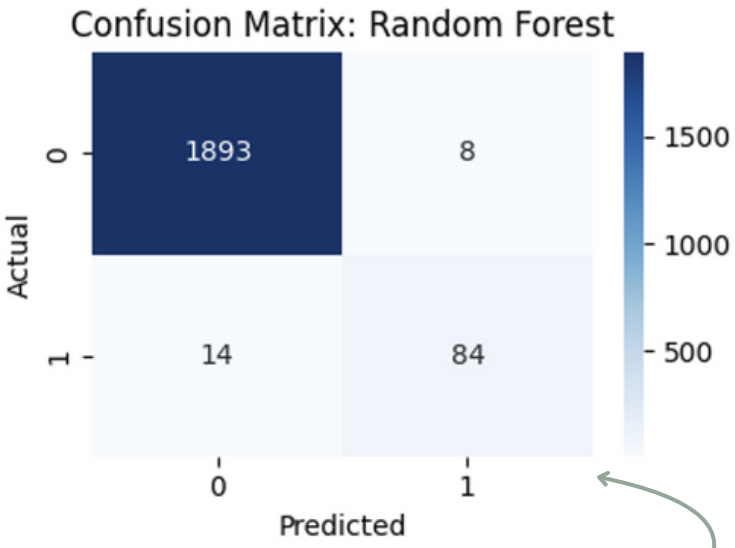
```
Training Decision Tree...
Rationale: Captures non-linear relationships and interactions between features.
Best Params: {'model__min_samples_split': 10, 'model__max_depth': 8}
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1901
           1       0.60      0.89      0.72        98

    accuracy                           0.97      1999
   macro avg       0.80      0.93      0.85      1999
weighted avg       0.97      0.97      0.97      1999
```

Precision for fraud is 0.60, recall 0.89, and F1-score 0.72. The model favors recall—important in fraud detection—while still maintaining good overall accuracy (97%). Decision Trees handle complex patterns and perform well here.

Random Forest



The ROC curve has an AUC of 0.97, showing excellent separation between fraud and non-fraud. The model performs consistently across all thresholds, with very few misclassifications.

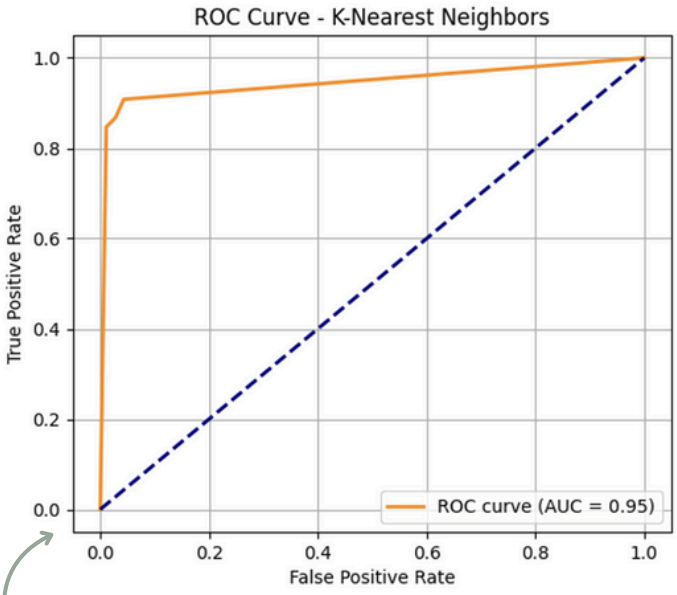


Out of 98 fraud cases, 84 were caught and 14 missed. Only 8 legitimate transactions were wrongly flagged as fraud. This balance shows the model is both precise and sensitive.

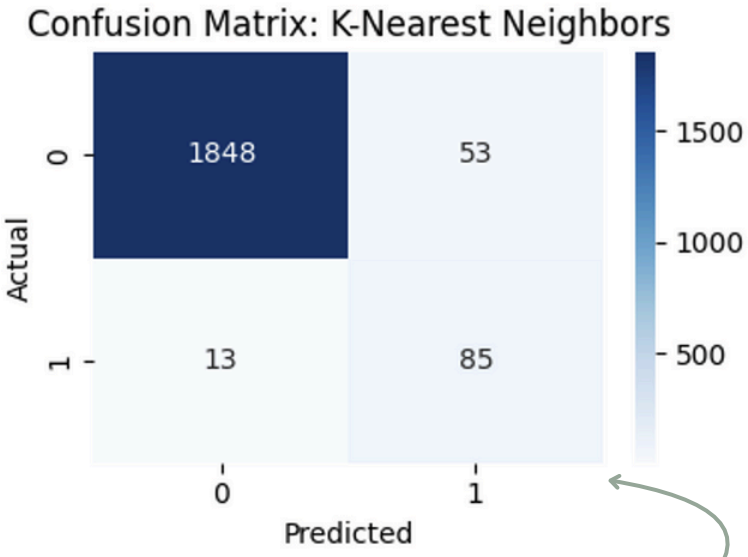
```
Training Random Forest...
Rationale: Ensemble method that improves accuracy and reduces variance.
Best Params: {'model__n_estimators': 100, 'model__min_samples_leaf': 2, 'model__max_depth': 8}
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1901
           1       0.91      0.86      0.88        98

    accuracy                           0.99      1999
   macro avg       0.95      0.93      0.94      1999
weighted avg       0.99      0.99      0.99      1999
```

Precision is 0.91, recall 0.86, and F1-score 0.88 for fraud. This is the best-performing model so far, with a strong balance between catching fraud and avoiding false alerts. Accuracy reaches 99% overall.

KNN



The ROC curve shows good separation with an AUC of 0.95, indicating strong model performance. The curve stays close to the top-left, meaning most frauds are caught with limited false positives.



The model detected 85 frauds out of 98, missing 13, and flagged 53 normal transactions as fraud. It maintains a high recall, which is valuable in fraud detection.

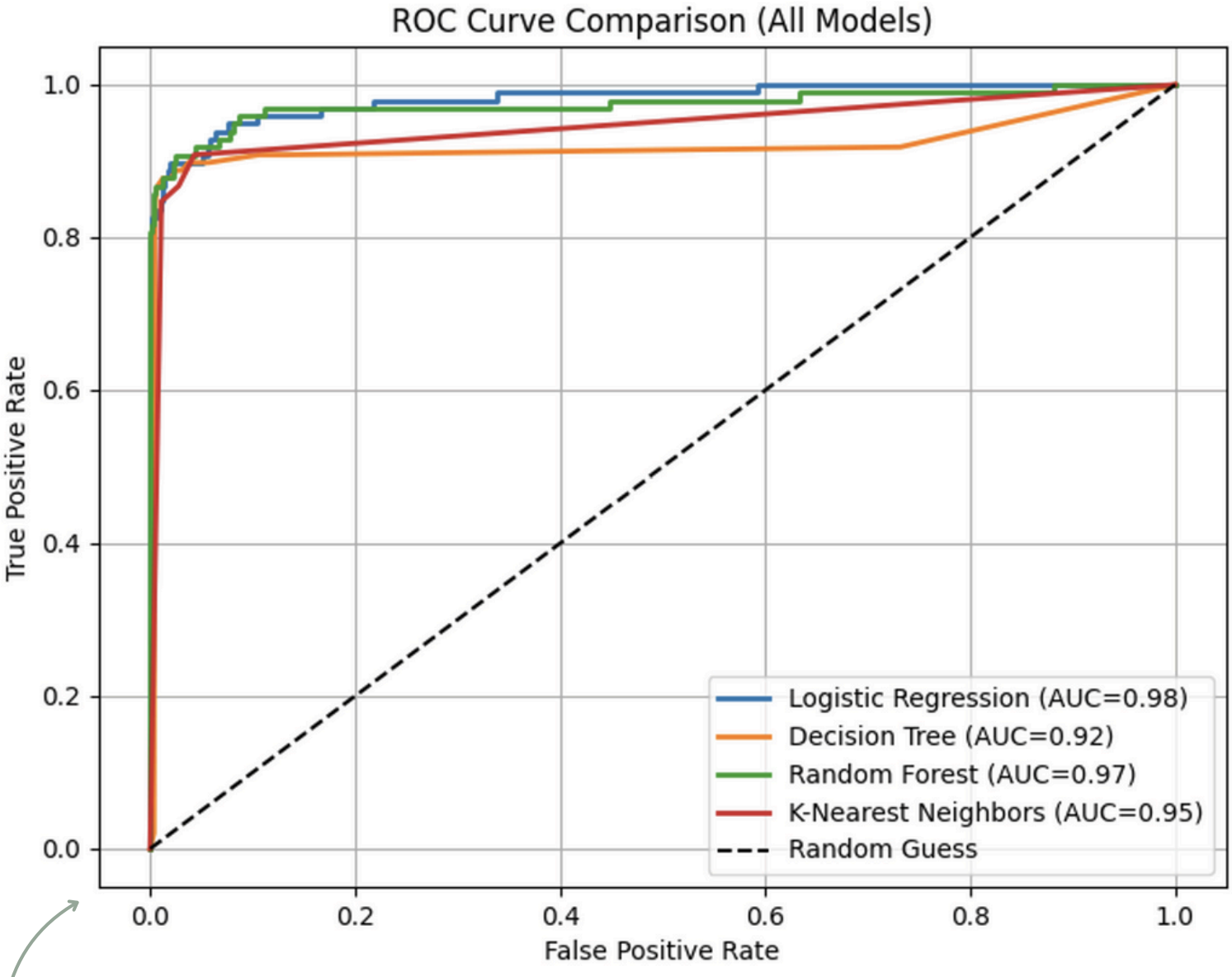
```
Training K-Nearest Neighbors...
Rationale: A distance-based approach effective on normalized data.
Best Params: {'model__n_neighbors': 3}
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1901
           1       0.62      0.87      0.72        98

    accuracy                           0.97      1999
   macro avg       0.80      0.92      0.85      1999
weighted avg       0.97      0.97      0.97      1999
```

Precision is 0.62, recall 0.87, and F1-score 0.72 for fraud. While not the most precise, it successfully identifies most frauds. Accuracy is 97%, showing reliable overall performance.

Final Evaluation Summary:

	Model	F1 Score	Precision	Recall	AUC
2	Random Forest	0.884211	0.913043	0.857143	0.974154
0	Logistic Regression	0.739496	0.628571	0.897959	0.98139
3	K-Nearest Neighbors	0.720339	0.615942	0.867347	0.945576
1	Decision Tree	0.719008	0.604167	0.887755	0.919927

Random Forest stands out with the best F1-score (0.88) and a strong balance of precision and recall. Logistic Regression follows closely with the highest AUC (0.98) but lower precision. All models perform well, but Random Forest offers the best overall trade-off.



All models show good separation ability, with curves hugging the top-left. Logistic Regression leads in AUC, but Random Forest maintains consistently high performance. The visual confirms that all models are effective, but Random Forest is the most reliable overall.

04

INSIGHTS

Insights, Recommendations & Future

From our results, we found several important takeaways. First, Random Forest clearly gave the best performance overall. It had high precision, meaning when it predicted a transaction was fraud, it was usually right. It also had strong recall, meaning it caught most frauds. This is a great balance for real-world fraud detection.

Another important finding was about using SMOTE. Without it, most models were biased toward the normal transactions. After we added synthetic fraud examples with SMOTE, recall improved for every model, and results became more stable and easier to repeat.

We also noticed that when we ran the training process multiple times, the model's precision sometimes improved a bit. This might be because SMOTE and hyperparameter tuning include randomness, so each run slightly changes the training data or model settings.

From a business point of view, we recommend using the Random Forest model in production, maybe through an API or as part of a batch system that checks transactions in real time.

It's also important to set up a monitoring pipeline. This will track the model's performance over time and alert if results start to drop. It should also allow for automatic retraining using new data, because fraud tactics are always changing.

Another useful idea is to adjust the decision threshold because the model's sensitivity can be tuned. For example, lowering the threshold might catch more frauds but will also cause more false positives.

Of course, the project had some limitations. The features in the dataset were anonymized, so we couldn't fully understand which patterns the model used. This limits how much we can interpret the model's decisions.

Also, the dataset is a bit old, and may not reflect how fraud looks today. So the model might not work perfectly on newer transactions without updates or retraining.

For future work, we recommend a few directions. Adding extra features like customer history, location data, or device information could make the model even stronger.

Finally, using unsupervised learning to detect new types of fraud that weren't seen in the training data could be a powerful next step.

```
KEY INSIGHTS:

- Random Forest consistently performs best in terms of balanced F1, precision, and recall.
- K-Nearest Neighbors, although simple, shows acceptable performance with scaled data.
- SMOTE proves effective in addressing class imbalance.
- All models demonstrate strong AUC values, suggesting good separation capability.


RECOMMENDATIONS:

- Deploy Random Forest in a production pipeline with appropriate monitoring.
- Periodically retrain models to reflect evolving fraud patterns.
- Consider cost-sensitive classification and threshold tuning for better business alignment.


LIMITATIONS & FUTURE WORK:

- Feature names are anonymized, limiting interpretability.
- Dataset is dated and may not reflect recent fraud techniques.
- Future work may explore deep learning or unsupervised anomaly detection methods.
```

05

CONCLUSION

Conclusion

To sum up, this project successfully applied machine learning methods to the real and serious issue of credit card fraud detection. Through careful data preparation, smart balancing techniques, and solid model training and testing, we found that Random Forest was the most effective model for spotting fraud in our dataset.

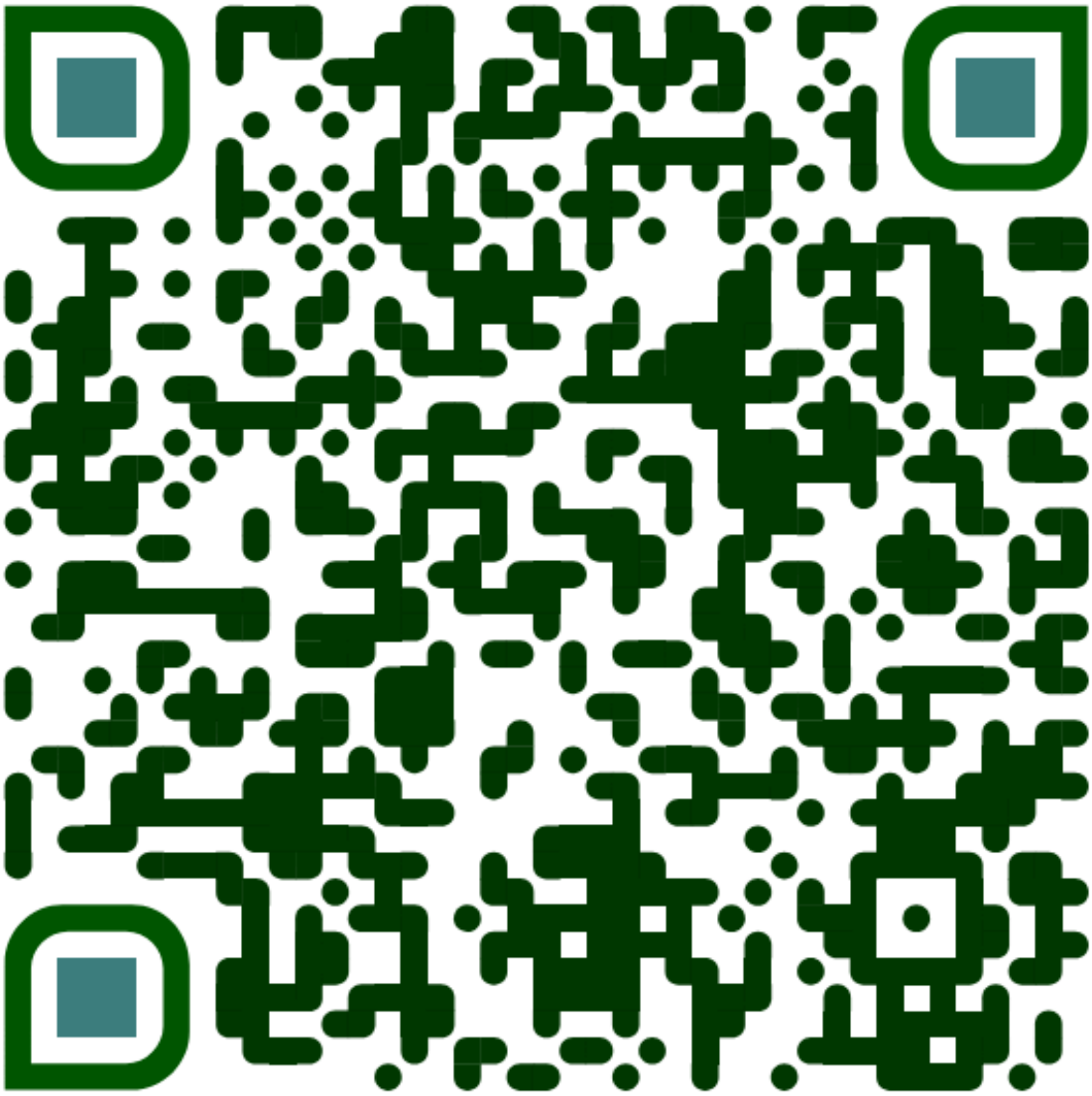
Beyond accuracy, we also focused on practical aspects like how to deploy the model, retrain it when needed, and adjust its sensitivity depending on the business situation. These points are key for using the model in real systems.

Even though our work had some limits it still gives a strong base for future development. It shows that with the right steps, machine learning can play a big role in stopping fraud.

We think that our report covers all the key parts of a complete ML project: choosing and cleaning data, building models, testing them, and making smart suggestions for the future.

06

APPENDIX



Our Github Project

End.