

CREDIT CARD DETECTION

Author: Badal Singh

Email: singh.badal3375@gmail.com

▲ *These are synthetic features (not real-world transaction fields like amount, time, etc.), because the demo model was trained on synthetic data. In a real project, you would replace these with actual transaction features (e.g., PCA components or domain-specific features).*

```
# Credit Card Detection using Machine Learning
```

```
This project focuses on detecting potential fraudulent credit card transactions using **M
```

```
---
```

```
### ▯ Features
```

- Data preprocessing and cleaning with **Pandas** and **NumPy**
- Data visualization with **Matplotlib** and **Seaborn**
- Feature extraction/engineering and text/vector transformations
- Machine learning model training using **Scikit-learn**
- Model saving and loading with **Pickle**
- Interactive **Streamlit** web app for real-time predictions

```
---
```

```
### ▯ Tech Stack
```

- **Programming Language**: Python 3.x
- **Libraries**:
 - Pandas (data manipulation)
 - NumPy (numerical computation)
 - Matplotlib & Seaborn (visualization)
 - Scikit-learn (ML algorithms, evaluation metrics)
 - Pickle (model saving/loading)
 - Vectorizers (feature transformation)
- **Frontend/UI**: Streamlit

```
---
```

```
### ▯ Project Structure
```

credit-card-detection/

- |
- |— data/ # Dataset files (CSV or others)
- |— notebooks/ # Jupyter notebooks for exploration
- |— models/ # Saved pickle model files
- |— src/ # Source code for preprocessing & training
 - | |— preprocess.py # Data cleaning & feature engineering
 - | |— train_model.py # ML training script
 - | |— evaluate_model.py # Model evaluation
- |— app.py # Streamlit web application
- |— requirements.txt # Dependencies
- |— README.md # Project documentation

🛠️ Installation & Setup

1. **Clone the repository**

```
git clone https://github.com/your-username/credit-card-detection_1.git
cd credit-card-detection
```

2. **Create virtual environment & install dependencies**

```
pip install -r requirements.txt
```

3. **Run the Streamlit app**

```
streamlit run app.py
```

🔄 Workflow

1. **Data Preprocessing & Cleaning**

- Handle missing values, duplicates, and scaling.
- Convert categorical data using encoders/vectorizers.

2. **Exploratory Data Analysis (EDA)**

- Visualize distributions, correlations, and patterns using Seaborn & Matplotlib.

3. **Model Training & Evaluation**

- Use ML algorithms from Scikit-learn (Logistic Regression, Random Forest, etc.).
- Evaluate with accuracy, precision, recall, F1-score, ROC curve, etc.

```

4. Model Saving & Deployment
- Save trained models with Pickle.
- Connect with Streamlit frontend for real-time predictions.

---

Frontend (Streamlit)
The Streamlit app provides a user-friendly web interface where users can:
- Input transaction details.
- Get real-time fraud detection results.
- Visualize prediction probabilities and model performance.

---

Future Improvements
- Integration with deep learning methods (e.g., TensorFlow/PyTorch).
- Deployment on cloud platforms (Heroku, AWS, GCP).
- Real-time API integration for production use.

---

Contributing
Contributions are welcome! Please fork the repo and submit a pull request.

---

License
This project is open-source under the MIT License.

---

requirements.txt

```

```

pandas2.2.2
numpy1.26.4
matplotlib3.9.2
seaborn0.13.2
scikit-learn1.5.1
streamlit1.38.0
pickle-mixin==1.0.2

```

If you use **Jupyter notebooks**, also add:

```

notebook
ipykernel

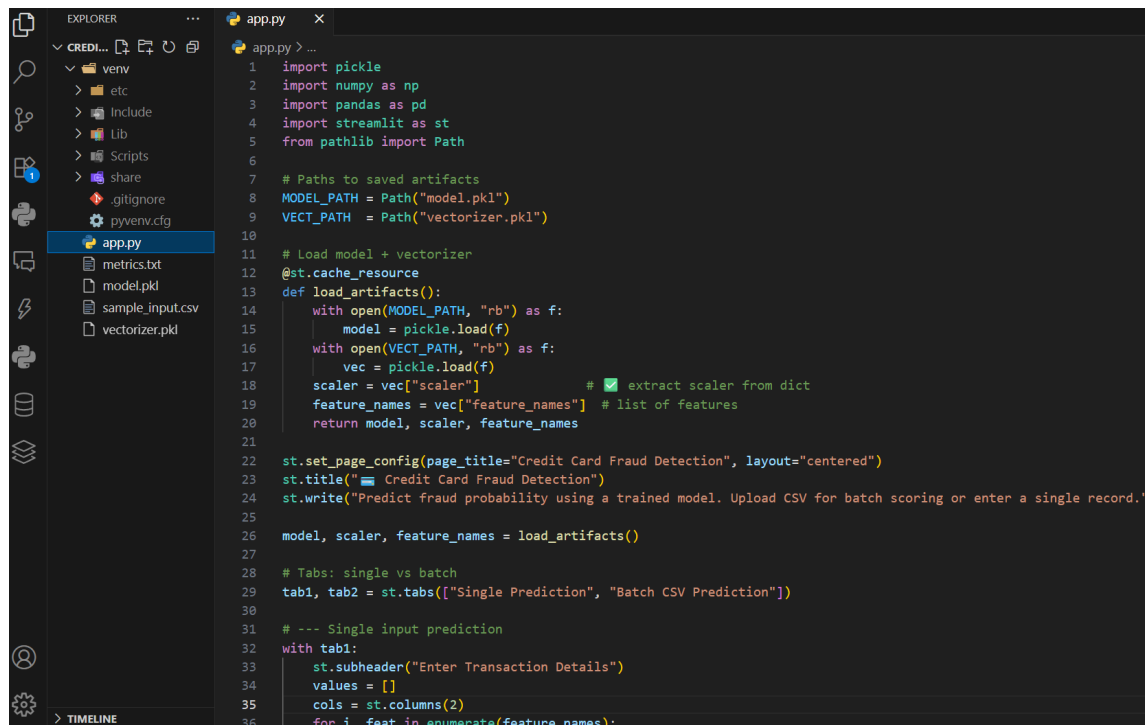
```

To Convert to PDF:

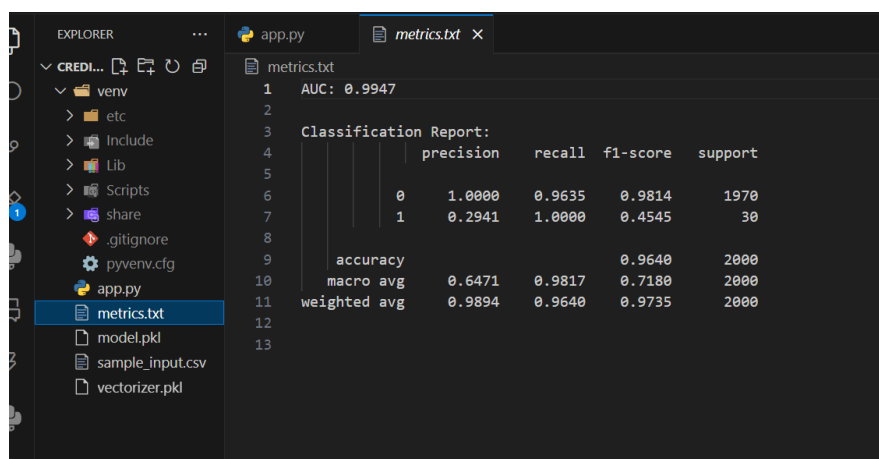
Save this content to a file named README.md, then run

```
pip install markdown-pdf
python -c "from markdown_pdf import MarkdownPdf, Section; pdf=MarkdownPdf(); pdf.add_section"
```

This will produce a high-quality PDF with your author information at the top.



```
1 import pickle
2 import numpy as np
3 import pandas as pd
4 import streamlit as st
5 from pathlib import Path
6
7 # Paths to saved artifacts
8 MODEL_PATH = Path("model.pkl")
9 VECT_PATH = Path("vectorizer.pkl")
10
11 # Load model + vectorizer
12 @st.cache_resource
13 def load_artifacts():
14     with open(MODEL_PATH, "rb") as f:
15         model = pickle.load(f)
16     with open(VECT_PATH, "rb") as f:
17         vec = pickle.load(f)
18     scaler = vec["scaler"] # extract scaler from dict
19     feature_names = vec["feature_names"] # list of features
20     return model, scaler, feature_names
21
22 st.set_page_config(page_title="Credit Card Fraud Detection", layout="centered")
23 st.title("Credit Card Fraud Detection")
24 st.write("Predict fraud probability using a trained model. Upload CSV for batch scoring or enter a single record.")
25
26 model, scaler, feature_names = load_artifacts()
27
28 # Tabs: single vs batch
29 tab1, tab2 = st.tabs(["Single Prediction", "Batch CSV Prediction"])
30
31 # --- Single input prediction
32 with tab1:
33     st.subheader("Enter Transaction Details")
34     values = []
35     cols = st.columns(2)
36     for i, feat in enumerate(feature_names):
```



```
1 AUC: 0.9947
2
3 Classification Report:
4 precision recall f1-score support
5
6 0 1.0000 0.9635 0.9814 1970
7 1 0.2941 1.0000 0.4545 30
8
9 accuracy 0.9640 2000
10 macro avg 0.6471 0.9817 0.7180 2000
11 weighted avg 0.9894 0.9640 0.9735 2000
12
13
```

```
sample_input.csv > data
1 f1,f2,f3,f4,f5,f6,f7,f8,f9,f10
2 -0.45913842391192117,-0.32464981622356626,0.7653338370565413,-0.12302948659856268,0.7106523527615226,-0.1067532300632265,-1.
3 0.3350651369812533,0.5860682766233394,-1.2188187343695995,1.6868411168553121,-0.9425592468015406,-0.2452442049831518,0.66747
4 0.3236759982592827,0.39381632993772,1.3187813251601583,0.5721970357234402,0.4806097419397185,-1.7425113998895798,1.053496876
5 0.16860402537731123,1.6486538232168038,0.1420881400688716,-0.40564300658418057,-1.1256344307999036,2.1383071262832125,0.0766
6 -2.9047825711902044,-1.7992324723651598,-0.06585608179988202,-0.4219484877018717,0.40791729424167034,-0.4251593145240789,0.5
7 0.5537346240227772,-0.3055803150502574,1.3523010093959922,0.753593771878911,1.1374452807333613,-1.8565691072615604,0.3433399
8 -0.9197513633505423,-0.9357959657603588,1.1277299214817833,0.3672586084350995,-0.6961116789791948,-0.559929936262654,-0.8136
9 -0.1619880973538309,-1.3538089096144361,-0.313543936890042,-2.0698675723746955,-0.03008750879349531,-0.759135871581265,-0.47
10 -0.43589190994268706,0.21221357062541288,-1.2189023633273262,0.29308154720856494,-3.2145897301528112,-0.0349191958707176,0.3
11 -0.535571576390779,0.32330085894294935,1.1847879139587878,0.35604334928,0.4119768898876125,1.5041076023762172,-0.65268699556
12
```

Final result

Credit Card Fraud Detection

Predict fraud probability using a trained model. Upload CSV for batch scoring or enter a single record.

Single Prediction Batch CSV Prediction

Enter Transaction Details

f1 0.3000	f2 0.1000
f3 -0.1000	f4 0.1000
f5 -0.4000	f6 0.3000
f7 -0.1000	f8 0.1000
f9 0.2000	f10 0.2000

Predict

Fraud Probability

0.0000

Prediction: ☒ Legit Transaction

Expected feature order: f1, f2, f3, f4, f5, f6, f7, f8, f9, f10