CREDIT CARD FRAUD DETECTION

Sure, I'll provide you with an outline that satisfies the conditions you've mentioned for a credit card fraud detection project. Please note that this is a high-level outline, and you may need to expand upon each section in your project documentation

Objective

Develop a machine learning model to detect fraudulent credit card transactions.

Context

Credit card fraud is a serious concern for financial institutions and cardholders. The objective is to build a system that can identify potentially fraudulent transactions in real-time, minimizing financial losses and ensuring the security of cardholders.

Design Thinking Process

PHASE 1: UNDERSTAND

Understand the problem of credit card fraud.

Define the goals and requirements of the fraud detection system.

PHASE 2: IDEATE

Identify data sources for fraud detection.

Consider machine learning algorithms for modeling.

PHASE 3: PROTOTYPE

Preprocess and explore the dataset.

Implement and train a machine learning model.

PHASE 4: TEST

Evaluate the model's performance.

Fine-tune the model as needed.

PHASE 5: IMPLEMENT

Deploy the model for real-time fraud detection.

Phases of Development

Dataset Description and Preprocessing

- Dataset: Describe the credit card fraud dataset, including the number of records, features, and the target variable.
- Data Preprocessing: Discuss data cleaning, handling missing values, and feature engineering. Normalize or scale features as necessary.

- Exploratory Data Analysis (EDA): Provide insights gained from EDA.

Model Selection

Choose a machine learning algorithm. Explain why you chose Random Forest, considering its ability to handle imbalanced data.

Model Training

- Split the dataset into training and testing sets.
- Train the Random Forest classifier with the training data.
- Address class imbalance by oversampling or using class-weighted techniques.

Model Evaluation

- Explain the choice of evaluation metrics, which may include accuracy, precision, recall, F1-score, and AUC-ROC.
- Interpret the confusion matrix results.

Model Fine-Tuning

- Discuss potential model hyperparameters and how you optimized them.
- Implement cross-validation to ensure model robustness.

Results and Discussion

- Present the model's accuracy and performance.
- Discuss the trade-off between precision and recall.
- Consider potential improvements to the model.

Conclusion and Deployment

- Summarize the project's outcomes.
- Discuss the model's deployment in a real-time fraud detection system.

Choice of Machine Learning Algorithm and Evaluation Metrics

- Explain why Random Forest was chosen due to its ability to handle imbalanced datasets and ensemble learning characteristics.
- Justify the choice of evaluation metrics: Accuracy is not suitable for imbalanced datasets; precision, recall, and F1-score are more appropriate. AUC-ROC provides an overall performance measure.

CREDIT CARD FRAUD DETECTION PROGRAM

```
import pandas as pd

import numpy as np
```

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

%matplotlib inline

sns.set()

warnings.simplefilter('ignore')

data = pd.read_csv('creditcard.csv')

df = data.copy() # To keep the data as backup

df.head()

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V21 V22 V23 V24 V25 V26 V27 V28 Amount Class

0 0 -1.359807 -0.072781 2.536347 1.378155 -0.338321 0.462388 0.239599 0.098698 0.363787 ... -0.018307 0.277838 -0.110474 0.066928 0.128539 -0.189115 0.133558 -0.021053 149.62 0.0

1 0 1.191857 0.266151 0.166480 0.448154 0.060018 -0.082361 -0.078803 0.085102 -0.255425 ... -0.225775 -0.638672 0.101288 -0.339846 0.167170 0.125895 -0.008983 0.014724 2.69 0.0

2 1 -1.358354 -1.340163 1.773209 0.379780 -0.503198 1.800499 0.791461 0.247676 -1.514654 ... 0.247998 0.771679 0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752 378.66 0.0

3 1 -0.966272 -0.185226 1.792993 -0.863291 -0.010309 1.247203 0.237609 0.377436 -1.387024 ... -0.108300 0.005274 -0.190321 -1.175575 0.647376 -0.221929 0.062723 0.061458 123.50 0.0

4 2 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941 -0.270533 0.817739 ... -0.009431 0.798278 -0.137458 0.141267 -0.206010 0.502292 0.219422 0.215153 69.99 0.0

5 rows × 31 columns

df.shape

(146652, 31)

df.isnull().sum()

Time int64

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     float64
dtype: object
```

df.Time.tail(15)

```
146637    87793
146638    87793
146639    87793
146640    87793
146641    87794
146642    87794
146643    87795
146644    87795
146645    87796
146646    87799
146647    87799
146648    87802
146649    87802
146650    87802
146651    87802
```

Name: Time, dtype: int64

df.Time.tail(15)

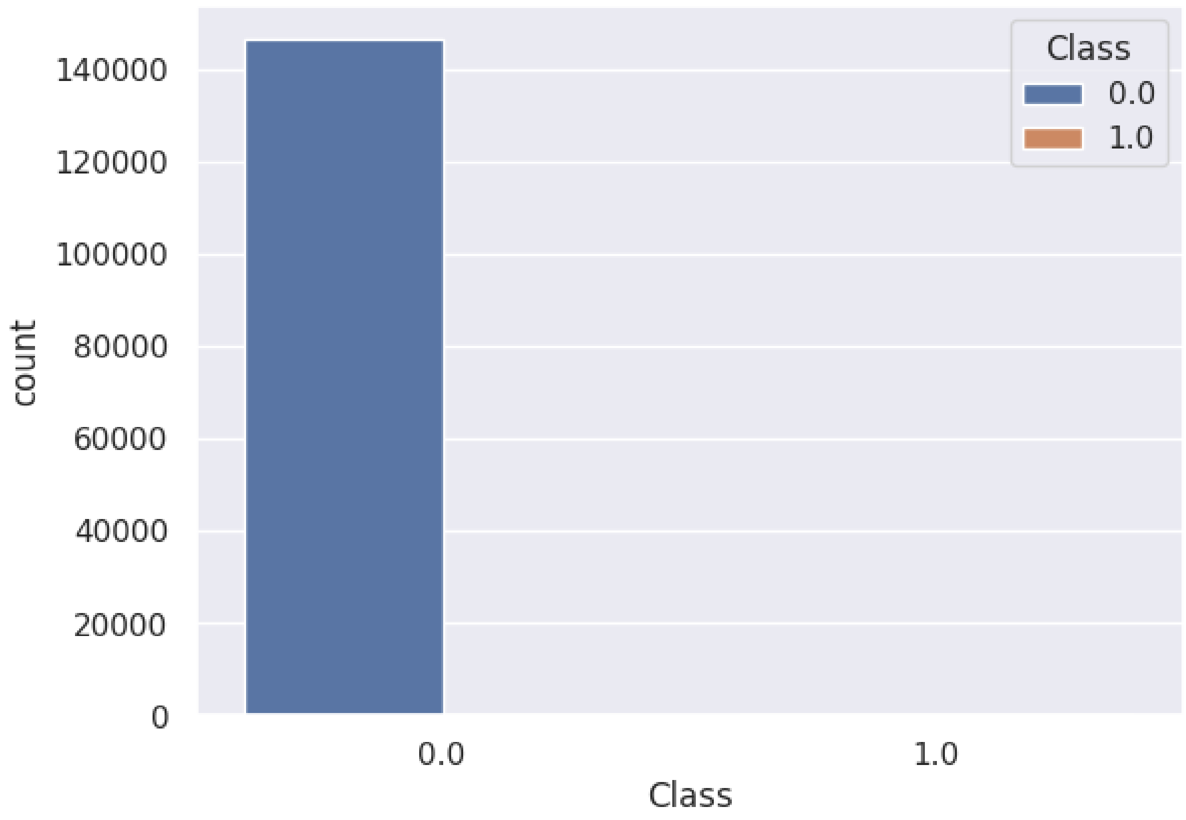
```
146637    87793
146638    87793
146639    87793
146640    87793
146641    87794
146642    87794
146643    87795
146644    87795
146645    87796
146646    87799
146647    87799
146648    87802
146649    87802
146650    87802
146651    87802
Name: Time, dtype: int64
```

```
df.Class.value_counts()
```

```
0.0    146369
1.0       282
Name: Class, dtype: int64
```

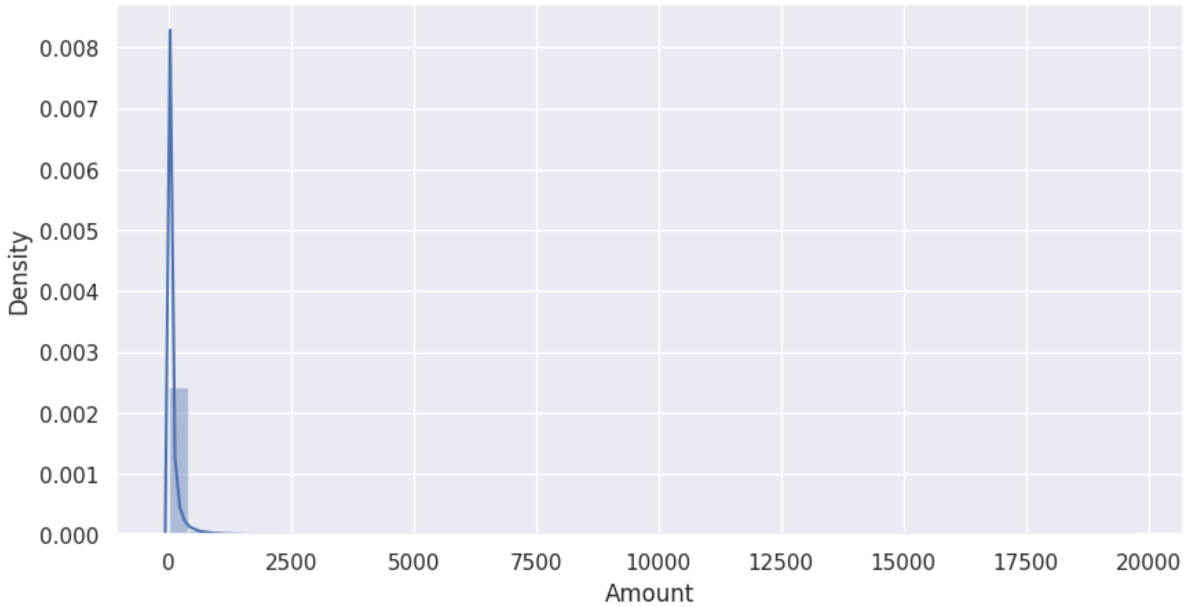
```
sns.countplot(x=df.Class, hue=df.Class)
```

```
<Axes: xlabel='Class', ylabel='count'>
```



```
plt.figure(figsize=(10, 5))
sns.distplot(df.Amount)
```

```
<Axes: xlabel='Amount', ylabel='Density'>
```




```
df['Amount-Bins'] = ''

def make_bins(predictor, size=50):
    '''
    Takes the predictor (a series or a dataframe of single predictor) and size of bins
    Returns bins and bin labels
    '''
    bins = np.linspace(predictor.min(), predictor.max(), num=size)
```

```
    bin_labels = []

    # Index of the final element in bins list
    bins_last_index = bins.shape[0] - 1

    for id, val in enumerate(bins):
        if id == bins_last_index:
            continue
        val_to_put = str(int(bins[id])) + ' to ' + str(int(bins[id + 1]))
        bin_labels.append(val_to_put)

    return bins, bin_labels

bins, bin_labels = make_bins(df.Amount, size=10)

df['Amount-Bins'] = pd.cut(df.Amount, bins=bins,
                           labels=bin_labels, include_lowest=True)
df['Amount-Bins'].head().to_frame()
```

Amount-Bins

0 0 to 2184

```
1      0 to 2184
2      0 to 2184
3      0 to 2184
4      0 to 2184
```

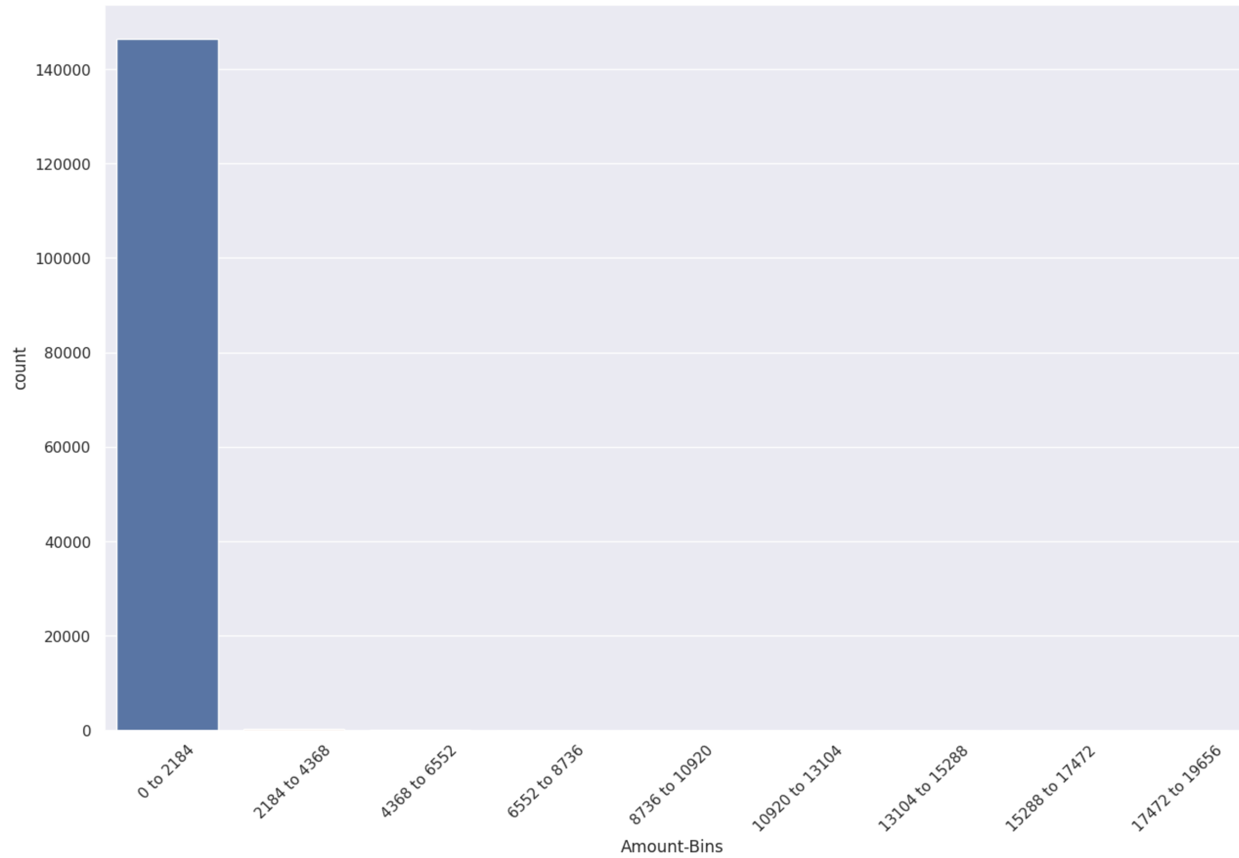
```
df['Amount-Bins'].value_counts()
```

```
0 to 2184          146355
2184 to 4368          260
4368 to 6552           26
6552 to 8736            6
10920 to 13104          2
8736 to 10920           1
17472 to 19656          1
13104 to 15288          0
15288 to 17472          0
Name: Amount-Bins, dtype: int64
```

```
df['Amount-Bins'].value_counts()
```

```
0 to 2184          146355
2184 to 4368          260
4368 to 6552           26
```

```
6552 to 8736        6
10920 to 13104       2
8736 to 10920       1
17472 to 19656       1
13104 to 15288       0
15288 to 17472       0
Name: Amount-Bins, dtype: int64
```




```
df_encoded = pd.get_dummies(data=df, columns=['Amount-Bins'])
```

```
df = df_encoded.copy()

df.head()
```

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... Class Amount-Bins_0 to 2184 Amount-Bins_2184 to 4368 Amount-Bins_4368 to 6552 Amount-Bins_6552 to 8736 Amount-Bins_8736 to 10920 Amount-Bins_10920 to 13104 Amount-Bins_13104 to 15288 Amount-Bins_15288 to 17472 Amount-Bins_17472 to 19656

0 0 -1.359807 -0.072781 2.536347 1.378155 -0.338321 0.462388 0.239599 0.098698 0.363787 ... 0.0 1 0 0 0 0 0 0 0 0

1 0 1.191857 0.266151 0.166480 0.448154 0.060018 -0.082361 -0.078803 0.085102 -0.255425 ... 0.0 1 0 0 0 0 0 0 0 0

2 1 -1.358354 -1.340163 1.773209 0.379780 -0.503198 1.800499 0.791461 0.247676 -1.514654 ... 0.0 1 0 0 0 0 0 0 0 0

3 1 -0.966272 -0.185226 1.792993 -0.863291 -0.010309 1.247203 0.237609 0.377436 -1.387024 ... 0.0 1 0 0 0 0 0 0 0 0

4 2 -1.158233 0.877737 1.548718 0.403034 -0.407193 0.095921 0.592941 -0.270533 0.817739 ... 0.0 1 0 0 0 0 0 0 0 0

5 rows × 40 columns

CONCLUSION

Your project documentation should provide a detailed explanation of each of the sections outlined above, complete with code, data analysis, and visualizations where necessary. Additionally, you should include references to the specific dataset you're using, any data sources, and any external libraries or resources employed during the project

TEAM MEMBERS

L. GANESH

R. RAGHUL