**Machine Learning Model Report**

**1. Introduction**

In this report, we detail the design choices made for the machine learning pipeline, evaluate the model’s performance, discuss potential improvements and future work, and provide the source code used in the project.

**2. Description of Design Choices**

**2.1 Data Preparation**

* **Data Source:** Describe the origin of your dataset (e.g., a public dataset, internal data).
* **Preprocessing Steps:** Outline the preprocessing steps, including data cleaning, feature selection, handling missing values, and scaling/normalization.
* **Handling Imbalance:** Detail the method used to address class imbalance (e.g., SMOTE, under-sampling).

**2.2 Model Selection**

* **Chosen Models:** Explain why specific models were chosen (e.g., Random Forest Classifier, Logistic Regression). Include considerations such as interpretability, performance, and computational efficiency.
* **Hyperparameter Tuning:** Describe the process used for hyperparameter tuning (e.g., grid search, random search).

**2.3 Evaluation Metrics**

* **Metrics Used:** Detail the metrics used to evaluate the model’s performance (e.g., accuracy, precision, recall, F1-score, ROC-AUC).
* **Cross-Validation:** Describe any cross-validation techniques used to ensure model robustness.

**3. Performance Evaluation**

**3.1 Results**

* **Training and Testing Performance:** Present the performance metrics for both training and testing datasets.
* **Confusion Matrix:** Include confusion matrices to show how well the model classifies each class.
* **Classification Report:** Provide detailed classification reports showing precision, recall, and F1-score for each class.

**3.2 Visualizations**

* **ROC Curve:** Include ROC curves if applicable.
* **Feature Importance:** Show feature importance plots if using models like RandomForest.

**3.3 Comparison**

* **Model Comparison:** Compare the performance of different models used. Explain which model performed best and why.

**4. Discussion of Future Work**

**4.1 Model Improvements**

* **Advanced Models:** Consider exploring more advanced models or algorithms.
* **Feature Engineering:** Discuss additional features that could be engineered to improve performance.

**4.2 Data Enhancements**

* **Additional Data:** Explore the potential for acquiring more data or additional data sources.
* **Data Quality:** Address any data quality issues that could be improved.

**4.3 Pipeline Enhancements**

* **Automation:** Discuss the potential for automating parts of the pipeline.
* **Scalability:** Consider how the pipeline could be scaled for larger datasets or different environments.

**5. Source Code**

**1. Load the Dataset**

#Load the dataset from a CSV file

df=pd.read\_csv("C:\Final\_capstone\creditcard.csv")

# Display the first few rows of the dataframe

print(df.head())s

df.dtypes

# Check for missing values

print("\nMissing Values Count:")

print(df.isnull().sum())

# Handle missing values (e.g., impute with mean)

df.fillna(df.mean(), inplace=True)

# Statistical summary

print("\nStatistical Summary:")

print(df.describe())

# Identify outliers (simple example using Z-score)

from scipy import stats

import numpy as np

# Calculate Z-scores

z\_scores = np.abs(stats.zscore(df.select\_dtypes(include=[np.number])))

outliers = (z\_scores > 3).all(axis=1)

print("\nNumber of outliers detected:", np.sum(outliers))

# Option 1: Fill missing values with the mean (or other strategies)

df.fillna(df.mean(), inplace=True)

# Option 2: Drop rows with missing values

df.dropna(inplace=True)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(df.select\_dtypes(include=[np.number]))

df\_scaled = pd.DataFrame(scaled\_features, columns=df.select\_dtypes(include=[np.number]).columns, index=df.index)

# Example of feature transformation

df['log\_Amount'] = np.log1p(df['Amount'])

# Example of feature creation

df['V1\_V2\_ratio'] = df['V1'] / (df['V2'] + 1e-8) # Avoid division by zero

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

import sklearn

print("scikit-learn version:", sklearn.\_\_version\_\_)

from imblearn.over\_sampling import SMOTE

X = df.drop(columns=['Class'])

y = df['Class']

smote = SMOTE()

X\_resampled, y\_resampled = smote.fit\_resample(X, y)

# Create a new DataFrame with resampled data

resampled\_df = pd.DataFrame(X\_resampled, columns=X.columns)

resampled\_df['Class'] = y\_resampled

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_resampled, y\_resampled, test\_size=0.3, random\_state=42)

# Initialize models

models = {

'Logistic Regression': LogisticRegression(),

'Random Forest': RandomForestClassifier(),

}

# Train and evaluate models

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

print(f"{name} Classification Report:")

print(classification\_report(y\_test, y\_pred))

# Train the chosen model (e.g., Random Forest)

best\_model = RandomForestClassifier()

best\_model.fit(X\_train, y\_train)

from sklearn.metrics import confusion\_matrix, roc\_auc\_score

y\_pred = best\_model.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(conf\_matrix)

roc\_auc = roc\_auc\_score(y\_test, best\_model.predict\_proba(X\_test)[:, 1])

print(f"ROC AUC Score: {roc\_auc}")

import joblib

# Save the model

joblib.dump(best\_model, 'model.pkl')

# Load the model for prediction

loaded\_model = joblib.load('model.pkl')