**Title: Predictive Analysis and Feature Selection for Failure Prediction using RandomForest Classifier**

**Abstract**

This project aims to predict failure in a dataset using advanced data preprocessing techniques, feature selection, and RandomForest classification. We utilized various data visualization techniques to explore the dataset, followed by selecting the top features impacting the target variable. Our model's performance was evaluated based on accuracy, classification reports, and feature importance.

**Introduction**

Predictive analytics involves using historical data, machine learning, and statistical algorithms to predict future outcomes. In this project, we focus on predicting failures using a machine learning approach. We leveraged the RandomForest classifier due to its robustness and ability to handle large datasets with higher dimensionality.

**Data Description**

The dataset was loaded from a CSV file and initially explored to understand its structure and contents. Various statistical summaries and visualizations were employed to identify data patterns and relationships among features.

**Data Preprocessing**

Data preprocessing involved handling missing values, scaling numeric features, and combining them with non-numeric features. We used the median value to fill missing numeric data to maintain the integrity of the dataset.

**Feature Selection**

We employed the SelectKBest method with ANOVA F-test to select the top 5 features most relevant to the target variable. This step helps in reducing dimensionality and improving model performance by focusing on the most impactful features.

**Model Training and Evaluation**

The dataset was split into training and testing sets in a 70:30 ratio. We trained a RandomForest classifier with 100 estimators and evaluated its performance using various metrics, including confusion matrix, classification report, and accuracy score. The model achieved satisfactory results, demonstrating its efficacy in predicting failures.

**Visualization**

We visualized the correlation matrix, target variable distribution, and relationships between features using pairplots. Additionally, feature importance was plotted to identify which features contributed most to the prediction model.

**Prediction Function**

A prediction function was developed to predict failures on new data samples. This function ensures that the input data matches the selected features and scales the data accordingly before making predictions.

**#CODE**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.feature\_selection import SelectKBest, f\_classif**

**from sklearn.ensemble import RandomForestClassifier**

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

**from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score**

**# Load the dataset**

**df = pd.read\_csv("data (1).csv")**

**# Initial Data Exploration**

**print(df.head())A screenshot of a computer

Description automatically generated**

**print(df.tail())A screenshot of a computer

Description automatically generated**

**print(df.shape) A screenshot of a computer

Description automatically generated**

**A screen shot of a computer code

Description automatically generatedprint(df.columns)**

**print(df.describe())A screenshot of a computer

Description automatically generated**

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

**A screenshot of a phone

Description automatically generated**

**# Fill NaN values for numeric columns only**

**numeric\_columns = df.select\_dtypes(include=['number']).columns**

**df[numeric\_columns] = df[numeric\_columns].fillna(df[numeric\_columns].median())**

**# Separate features and target**

**target\_column = 'Response' # Adjust this to your target column name**

**features = df.drop(columns=[target\_column])**

**target = df[target\_column]**

**# Apply StandardScaler only to numeric features**

**scaler = StandardScaler()**

**numeric\_features = features.select\_dtypes(include=['number'])**

**numeric\_features\_scaled = scaler.fit\_transform(numeric\_features)**

**# Combine scaled numeric features with non-numeric features**

**non\_numeric\_features = features.select\_dtypes(exclude=['number'])**

**df\_scaled = pd.DataFrame(numeric\_features\_scaled, columns=numeric\_features.columns)**

**df\_scaled = pd.concat([df\_scaled, non\_numeric\_features.reset\_index(drop=True)], axis=1)**

**# Add the target column back**

**df\_scaled[target\_column] = target.reset\_index(drop=True)**

**# Verify the resulting DataFrame**

**print(df\_scaled.head())**

**# Visualization: Correlation matrix**

**plt.figure(figsize=(12, 8))**

**sns.heatmap(df\_scaled.corr(), annot=True, cmap='coolwarm')**

**plt.title('Correlation Matrix')**

**plt.show()**

**# Visualization: Distribution of the target variable**

**sns.countplot(x=target\_column, data=df\_scaled)**

**plt.title('Target Variable Distribution'A chart of data on a white background

Description automatically generated)**

**plt.show()A screenshot of a computer screen

Description automatically generated**

**# Visualization: Pairplot to visualize relationships**

**sns.pairplot(df\_scaled, hue=target\_column, diag\_kind='kde')**

**plt.show()**

**A graph with blue squares

Description automatically generated**

**# Feature Selection: Selecting the top features**

**X = df\_scaled.drop(columns=[target\_column])**

**y = df\_scaled[target\_column]**

**selector = SelectKBest(score\_func=f\_classif, k=5)**

**X\_new = selector.fit\_transform(X, y)**

**# Displaying selected features**

**selected\_features = X.columns[selector.get\_support()]**

**print("Selected Features:", selected\_features)**

**# Splitting the data into training and testing sets**

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

**# Initializing and training the model**

**model = RandomForestClassifier(n\_estimators=100, random\_state=42)**

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

**# Predicting on the test set**

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

**# Model evaluation**

**print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**print("Classification Report:\n", classification\_report(y\_test, y\_pred))**

**print("Accuracy Score:", accuracy\_score(y\_test, y\_pred))**

**# Feature Importance**

**importances = model.feature\_importances\_**

**indices = importances.argsort()[::-1]**

**plt.figure(figsize=(10, 6))**

**plt.title("Feature Importances")**

**plt.bar(range(X\_new.shape[1]), importances[indices], align="center")**

**plt.xticks(range(X\_new.shape[1]), selected\_features[indices], rotation=45)**

**plt.show()**

**A graph with blue squares

Description automatically generated**

**# Prediction function**

**def predict\_failure(new\_data):**

**# Ensure new data has the correct number of features**

**if len(new\_data[0]) != len(selected\_features):**

**raise ValueError(f"Expected {len(selected\_features)} features but got {len(new\_data[0])}.")**

**# Create DataFrame with the new data**

**new\_data\_df = pd.DataFrame(new\_data, columns=selected\_features)**

**# Scaling the new data**

**new\_data\_scaled = scaler. transform(new\_data\_df)**

**# Predicting using the trained model**

**prediction = model.predict(new\_data\_scaled)**

**return prediction**

**# Example prediction**

**new\_sample = [[10, 2, 50, 0.5, 30]] # Ensure this matches the selected features**

**A screenshot of a computer

Description automatically generatedprint("Failure Prediction:", predict\_failure(new\_sample))**

**Conclusion**

The project successfully demonstrated the application of data preprocessing, feature selection, and machine learning in predicting failures. The RandomForest classifier proved effective, with significant insights gained from the feature importance analysis. Future work may involve exploring other models and further refining the preprocessing steps for even better performance.