DEBRE BERHAN UNIVERSITY
College of Computing
Department of Software Engineering

Individual Assignment

Course: Fundamentals of Machine Learning

Name: Abigiya Elias
ID : 1500002

Submitted To : Derbewu Felashman (MSC)

Financial Health Prediction - Documentation

1. Project Overview

The **Financial Health Prediction** project utilizes machine learning to predict an individual's financial health based on various financial features such as income, savings, expenditure, debt, and investment. The prediction is made using a Random Forest Classifier model that categorizes individuals as having either good or bad financial health.

The project is designed to help individuals assess their financial status and make informed decisions to improve their financial stability. After training and evaluating the model, it is deployed as a RESTful API using **FastAPI**, making it easy to access and use.

Key Features:

- **Data Exploration**: Understand the dataset and relationships between the features.
- **Data Preprocessing**: Prepare the data by splitting it into training/testing sets and scaling it.
- **Model Training**: Use a Random Forest Classifier to predict financial health.
- **Model Evaluation**: Assess the model using performance metrics like accuracy and classification reports.
- **FastAPI Deployment**: Deploy the model as an API to predict financial health based on user input.

Benefits:

- Real-time financial health predictions.
- Easy integration with web or mobile applications via the FastAPI RESTful interface.
- Provides actionable insights for individuals to manage their finances.

2. Data Exploration

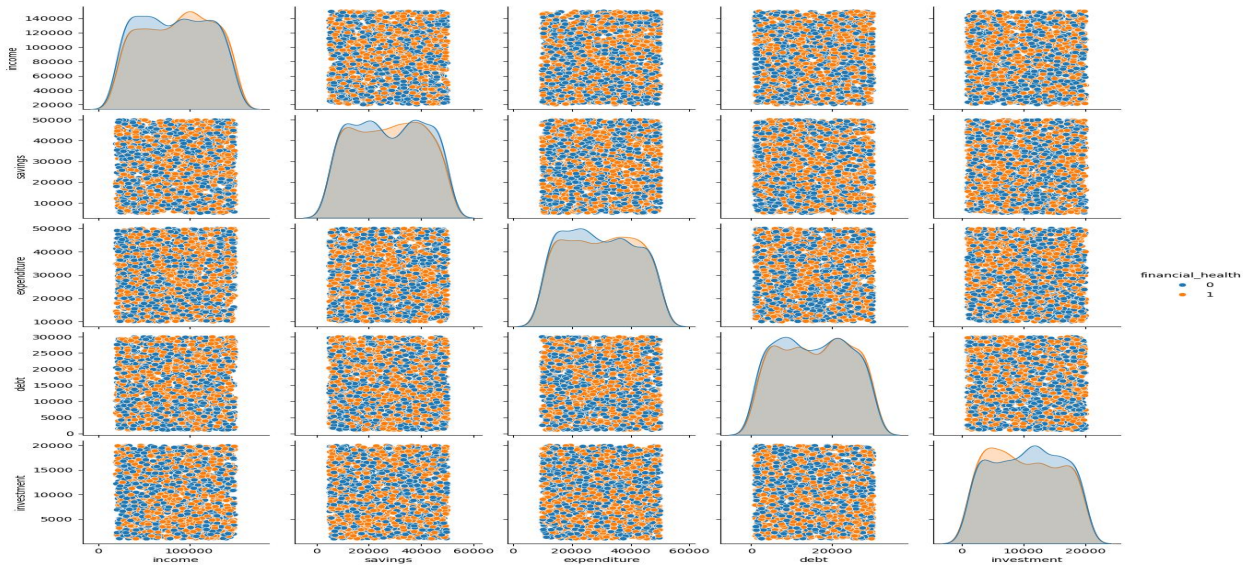
The project starts by loading the dataset and performing basic exploration.

python
```
# Load the dataset
df = pd.read_csv("/mnt/data/financial_data_large.csv")
# Data explorationprint(df.info())print(df.describe())
sns.pairplot(df, hue='financial_health')
plt.show()
```

Description:

- df.info(): Displays general information about the dataset (e.g., number of rows, columns, data types).
- df.describe(): Provides summary statistics for each feature in the dataset.

- **Pairplot Visualization**: Plots pairwise relationships between features and the target variable financial_health.



3. Data Preprocessing

This step prepares the data for machine learning by splitting it into features and the target variable, scaling the features, and dividing it into training and testing sets.

python

```
# Splitting features and target variable
X = df.drop(columns=['financial_health'])
y = df['financial_health']
# Splitting into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Description:

- **Splitting Data**: X represents the features (income, savings, expenditure, debt, investment), and y represents the target variable financial_health.
- **Training/Testing Split**: The data is split into 80% for training and 20% for testing.
- **Scaling**: The StandardScaler is used to standardize the features to have a mean of 0 and a standard deviation of 1.

4. Model Training

The model is trained using a Random Forest Classifier, which is a popular ensemble method known for its robustness and ability to handle various data complexities.

python
 # Model training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

Description:

- A **Random Forest Classifier** is initialized with 100 trees (estimators) and trained on the scaled training data (X_train_scaled and y_train).

5. Model Evaluation

The model's performance is evaluated using accuracy, classification report, and confusion matrix.

python
Model evaluation
y_pred = model.predict(X_test_scaled)print("Accuracy:", accuracy_score(y_test, y_pred))print(classification_report(y_test, y_pred))

Description:

- **Accuracy**: The overall accuracy of the model on the test set is printed.
- **Classification Report**: Provides precision, recall, and F1-score for each class.
- **Confusion Matrix**: A heatmap is generated to visualize the number of true positives, false positives, true negatives, and false negatives.

6. Confusion Matrix Visualization

A confusion matrix is plotted to visually evaluate the model's performance.

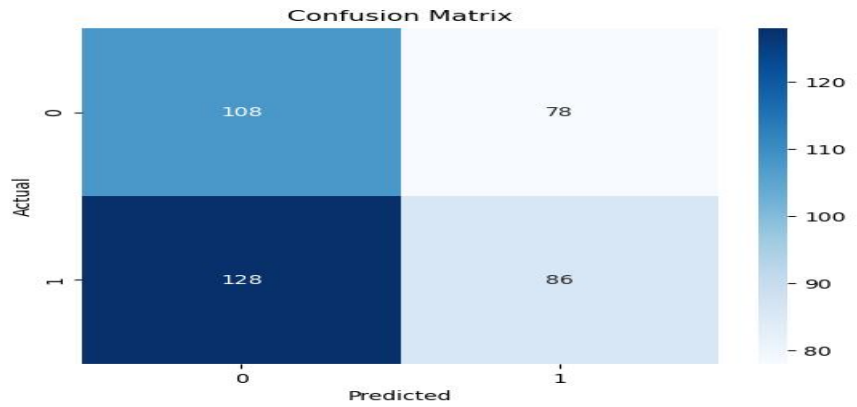
Python

 # Confusion matrix visualization
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")

plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

Description:

- This visualization provides a clear overview of how well the model is distinguishing between the different financial health classes (good vs. bad).



7. Saving the Model and Scaler

The trained model and scaler are saved using joblib so they can be used later for predictions or API deployment.

Python

Save the model and scaler
joblib.dump(model, "financial_health_model.pkl")
joblib.dump(scaler, "scaler.pkl")

Description:

- financial_health_model.pkl: The trained Random Forest Classifier model is serialized and saved.
- scaler.pkl: The scaler object used to standardize features is saved for future use.

8. Making Predictions on New Data

A function is created to make predictions on new input data by applying the saved scaler and model.

Python

 # Making predictions on new datadef predict_financial_health(new_data):
new_data_df = pd.DataFrame([new_data], columns=X.columns)
new_data_scaled = scaler.transform(new_data_df)

```
    prediction = model.predict(new_data_scaled)
    return prediction[0]
# Example prediction
sample_data = [50000, 10000, 20000, 5000, 7000]print("Predicted financial health:",
predict_financial_health(sample_data))
```

Description:

- The function predict_financial_health takes in new financial data (e.g., income, savings, expenditure, debt, and investment), scales it using the saved scaler, and returns the predicted financial health.

9. Deploying the Model with FastAPI

The trained model and scaler are deployed as a RESTful API using **FastAPI**, which allows users to submit their financial data and receive predictions.

Python

```
from fastapi import FastAPIimport joblibimport pandas as pdimport numpy as
npfrom pydantic import BaseModel
# Load the trained model and scaler
model = joblib.load("financial_health_model.pkl")
scaler = joblib.load("scaler.pkl")
# Define feature names (Ensure these match your dataset)
feature_names = ["income", "savings", "expenditure", "debt", "investment"]
# Initialize FastAPI
app = FastAPI()
# Define input data structureclass FinancialData(BaseModel):
    income: float
    savings: float
    expenditure: float
    debt: float
    investment: float
@app.post("/predict")def predict(data: FinancialData):
    try:
        # Convert JSON input to DataFrame
        new_data_df = pd.DataFrame([data.dict().values()], columns=feature_names)

        # Scale input data
        new_data_scaled = scaler.transform(new_data_df)

        # Make prediction
        prediction = model.predict(new_data_scaled)[0]

        return {"financial_health_prediction": int(prediction)}
```

```
except Exception as e:
    return {"error": str(e)}
```

Description:

- **FastAPI**: A simple web framework that allows easy deployment of the model as an API.
- **Pydantic**: Used to validate the input data format (JSON) and convert it to a DataFrame.
- **POST** /predict **Endpoint**: Accepts financial data via a POST request, scales it, and returns the prediction.

10. Running the API

To run the FastAPI application and start the server:

uvicorn api:app --reload

Description:

- The **FastAPI** server is launched, and the API can now be accessed for making predictions.

Conclusion

The **Financial Health Prediction** project leverages machine learning to predict an individual's financial health, and it is deployed as an API for real-time predictions. It helps individuals gain insights into their financial situation and make informed decisions to improve their financial stability. The solution is scalable, easy to use, and accessible for integration into various applications.