LightPDF

Heart Disease Project — Detailed Report

1. Step 1:Data Collection And Load

This section titled 'Step 1:Data Collection And Load' focuses on a specific step in the project and explains what we did and why. I wrote code to perform the main actions described here; the steps are concise and practical. The aim is to make the dataset ready for modeling and to produce clear outputs that a reviewer can follow. Following these steps ensures the work is reproducible and easy to explain during submission. If you want extra detail for any sub-step, I can expand it into more lines or add comments in the code.

2. Step 2: Data Preprocessing

This section titled 'Step 2: Data Preprocessing' focuses on a specific step in the project and explains what we did and why. I wrote code to perform the main actions described here; the steps are concise and practical. The aim is to make the dataset ready for modeling and to produce clear outputs that a reviewer can follow. Following these steps ensures the work is reproducible and easy to explain during submission. If you want extra detail for any sub-step, I can expand it into more lines or add comments in the code.

3. Univariate Analysis

This section titled 'Univariate Analysis' focuses on a specific step in the project and explains what we did and why. I wrote code to perform the main actions described here; the steps are concise and practical. The aim is to make the dataset ready for modeling and to produce clear outputs that a reviewer can follow. Following these steps ensures the work is reproducible and easy to explain during submission. If you want extra detail for any sub-step, I can expand it into more lines or add comments in the code.

4. Bivariate Analysis

This section titled 'Bivariate Analysis' focuses on a specific step in the project and explains what we did and why. I wrote code to perform the main actions described here; the steps are concise and practical. The aim is to make the dataset ready for modeling and to produce clear outputs that a reviewer can follow. Following these steps ensures the work is reproducible and easy to explain during submission. If you want extra detail for any sub-step, I can expand it into more lines or add comments in the code.

5. Step 3: Data Splitting

This section titled 'Step 3: Data Splitting' focuses on a specific step in the project and explains what we did and why. I wrote code to perform the main actions described here; the steps are concise and practical. The aim is to make the dataset ready for modeling and to produce clear outputs that a reviewer can follow. Following these steps ensures the work is reproducible and easy to explain during submission. If you want extra detail for any sub-step, I can expand it into more lines or add comments in the code.

code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pickle
from sklearn.preprocessing import LabelEncoder
```

LightPDF

Code — Step 1:Data Collection And Load

Below is the code I wrote in the notebook for this step. I kept variable names clear and the logic straightforward.

```
class CarPriceDataset:
    def __init__(self, filepath):
        self.filepath = filepath
        self.df = None



    def load_data(self):
        self.df = pd.read_csv(self.filepath)
        return self.df

    def initial_info(self):
        return self.df.info(), self.df.head()


dataset = CarPriceDataset("cleaned_heart_.csv")

df = dataset.load_data()

info, head = dataset.initial_info()

print(" Dataset Info:")
print(info)

print("\n First 5 Rows:")
print(head)
```

Code — Step 2: Data Preprocessing

Below is the code I wrote in the notebook for this step. I kept variable names clear and the logic straightforward.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler

class DataPreprocessing:
    def __init__(self, df):
        self.df = df.copy()

    def check_missing(self):
        """Check missing values"""
        print("\n Missing Values in Dataset:")
        print(self.df.isnull().sum())

    def handle_missing(self):
        """Fill or drop missing values"""
        self.df = self.df.dropna()
        return self.df

    def remove_duplicates(self):
        """Remove duplicate rows"""
        before = self.df.shape[0]
        self.df = self.df.drop_duplicates()
        after = self.df.shape[0]
        print(f"\nRemoved {before - after} duplicate rows.")
        return self.df

    def encode_categorical(self):
        """Encode categorical columns if any"""
        le = LabelEncoder()
        for col in self.df.select_dtypes(include=['object']).columns:
            self.df[col] = le.fit_transform(self.df[col])
        return self.df

    def scale_features(self):
        """Scale numerical features"""
        scaler = StandardScaler()
        num_cols = self.df.select_dtypes(include=['int64', 'float64']).columns
        self.df[num_cols] = scaler.fit_transform(self.df[num_cols])
        return self.df
```

LightPDF

```
    def get_processed_data(self):
        return self.df


if __name__ == "__main__":
    df = pd.read_csv("cleaned_heart_.csv")

    preprocess = DataPreprocessing(df)

    preprocess.check_missing()
    df = preprocess.handle_missing()
    df = preprocess.remove_duplicates()
    df = preprocess.encode_categorical()
    df = preprocess.scale_features()

    print("\n Final Processed Data (first 5 rows):")
    print(df.head())
```

Code — Univariate Analysis

Below is the code I wrote in the notebook for this step. I kept variable names clear and the logic straightforward.

```
class UnivariateAnalysis:
    def __init__(self, df):
        self.df = df

    def analyze_numerical(self):
        """Plots histogram and boxplot for numerical features"""
        num_cols = self.df.select_dtypes(include=['int64', 'float64']).columns
        for col in num_cols:
            plt.figure(figsize=(12,5))

            # Histogram
            plt.subplot(1,2,1)
            sns.histplot(self.df[col], kde=True, bins=30)
            plt.title(f"Histogram of {col}")

            # Boxplot
            plt.subplot(1,2,2)
            sns.boxplot(x=self.df[col])
            plt.title(f"Boxplot of {col}")

            plt.show()

    def analyze_categorical(self):
        """Plots countplot for categorical features"""
        cat_cols = self.df.select_dtypes(include=['object']).columns
        for col in cat_cols:
            plt.figure(figsize=(6,4))
            sns.countplot(x=self.df[col])
            plt.title(f"Countplot of {col}")
            plt.show()


if __name__ == "__main__":
    df = pd.read_csv("cleaned_heart_.csv")

    uni = UnivariateAnalysis(df)

    print("\n ■ Numerical Features Analysis...")
    uni.analyze_numerical()

    print("\n ■ Categorical Features Analysis...")
    uni.analyze_categorical()
```

Code — Bivariate Analysis

Below is the code I wrote in the notebook for this step. I kept variable names clear and the logic straightforward.

```
class BivariateAnalysis:
    def __init__(self, df, target):
        self.df = df
        self.target = target
```

LightPDF

```
    def analyze_numerical(self):
        """Plots boxplot and scatterplot for numerical features vs target"""
        num_cols = self.df.select_dtypes(include=['int64', 'float64']).columns
        num_cols = [col for col in num_cols if col != self.target]

        for col in num_cols:
            plt.figure(figsize=(12,5))

            # Boxplot
            plt.subplot(1,2,1)
            sns.boxplot(x=self.df[self.target], y=self.df[col])
            plt.title(f"Boxplot of {col} vs {self.target}")

            # Scatterplot
            plt.subplot(1,2,2)
            sns.scatterplot(x=self.df[col], y=self.df[self.target], alpha=0.6)
            plt.title(f"Scatterplot of {col} vs {self.target}")

            plt.show()

    def analyze_categorical(self):
        """Plots countplot for categorical features vs target"""
        cat_cols = self.df.select_dtypes(include=['object']).columns

        for col in cat_cols:
            plt.figure(figsize=(7,5))
            sns.countplot(x=self.df[col], hue=self.df[self.target])
            plt.title(f"Countplot of {col} vs {self.target}")
            plt.show()


if __name__ == "__main__":
    df = pd.read_csv("cleaned_heart_.csv")


    target_col = "target"

    bi = BivariateAnalysis(df, target_col)

    print("\n ■ Numerical Features vs Target...")
    bi.analyze_numerical()

    print("\n ■ Categorical Features vs Target...")
    bi.analyze_categorical()
```

LightPDF

Code — Step 3: Data Splitting

Below is the code I wrote in the notebook for this step. I kept variable names clear and the logic straightforward.

```
class DataSplit:
    def __init__(self, df, target):
        self.df = df
        self.target = target
        self.X = None
        self.y = None
        self.X_train = None
        self.X_test = None
        self.y_train = None
        self.y_test = None

    def split_features_target(self):
        """Separate features and target"""
        self.X = self.df.drop(columns=[self.target])
        self.y = self.df[self.target]
        print("\n Features and Target separated.")
        return self.X, self.y

    def train_test_split(self, test_size=0.2, random_state=42):
        """Split dataset into train and test sets"""
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
            self.X, self.y, test_size=test_size, random_state=random_state
        )
        print(f"\n Data Split Done: Train = {self.X_train.shape[0]} rows, Test = {self.X_test.shape[0]} rows")
        return self.X_train, self.X_test, self.y_train, self.y_test
```

```
if __name__ == "__main__":
    df = pd.read_csv(r"cleaned_heart_.csv")

    target_col = "target"

    splitter = DataSplit(df, target_col)

    X, y = splitter.split_features_target()
    X_train, X_test, y_train, y_test = splitter.train_test_split(test_size=0.2, random_state=42)

    print("\n First 5 rows of Training Features:")
    print(X_train.head())
    print("\n First 5 rows of Training Labels:")
    print(y_train.head())
```

LightPDF