Name : Ch Mubashir

SAP : 56892.

Lab Task 03 :

Data Preprocessing with heart Dataset.

Lab Tasks

1. **Separate features and target :**

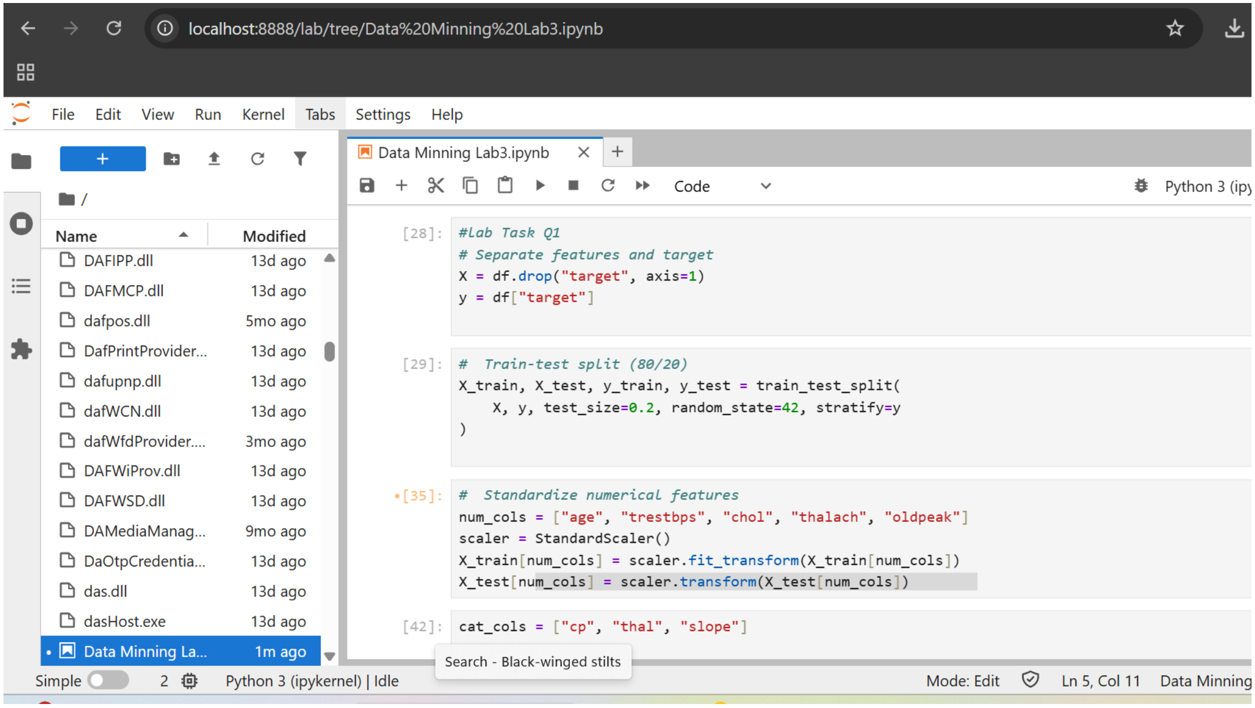
 Target variable: target (1 = disease, 0 = no disease).

2. **Split the data.**

3. **Standardize numerical features :**

 Columns: age, trestbps, chol, thalach, oldpeak.

 Use StandardScaler, fit on training only, transform both train and test



```
[28]: #lab Task Q1
      # Separate features and target
      X = df.drop("target", axis=1)
      y = df["target"]
```

```
[29]: #  Train-test split (80/20)
      X_train, X_test, y_train, y_test = train_test_split(
          X, y, test_size=0.2, random_state=42, stratify=y
      )
```

```
•[35]: #  Standardize numerical features
       num_cols = ["age", "trestbps", "chol", "thalach", "oldpeak"]
       scaler = StandardScaler()
       X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
       X_test[num_cols] = scaler.transform(X_test[num_cols])
```

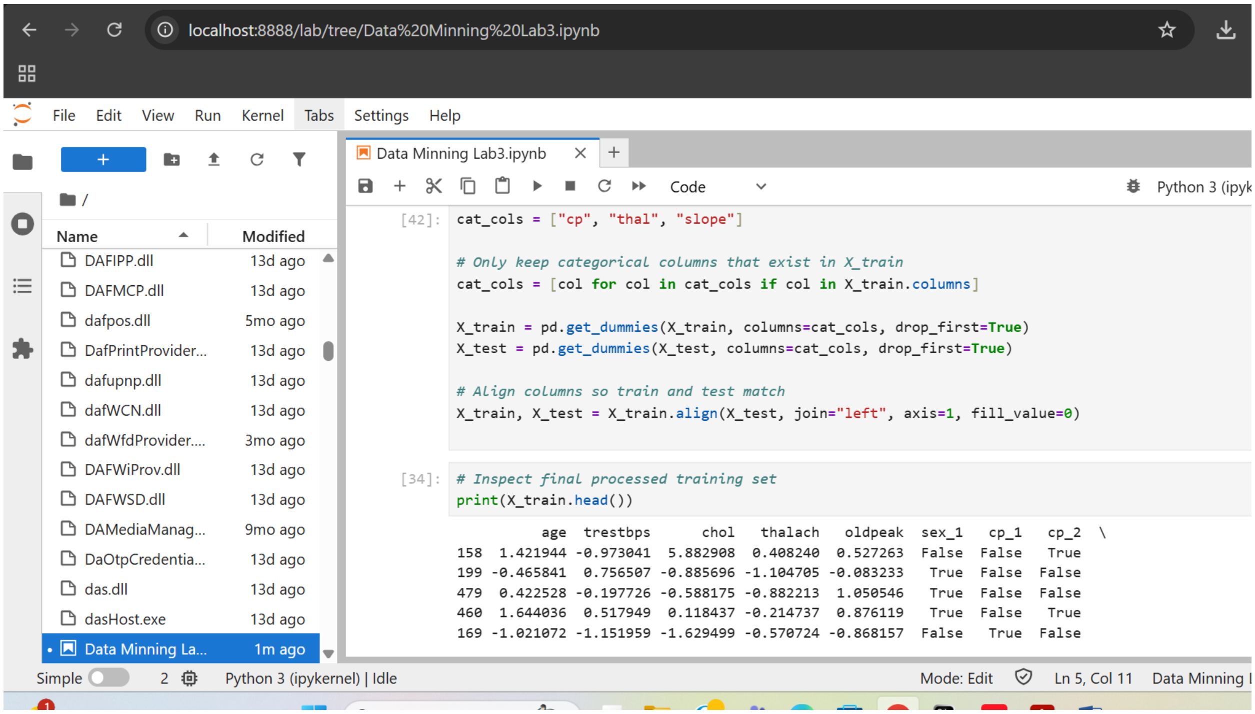
```
[42]: cat_cols = ["cp", "thal", "slope"]
```

4. **One-Hot Encode categorical features**

 Columns: cp, thal, slope.

 Use pd.get_dummies(..., drop_first=True).

5. **Check processed training data :**



```
[42]: cat_cols = ["cp", "thal", "slope"]

      # Only keep categorical columns that exist in X_train
      cat_cols = [col for col in cat_cols if col in X_train.columns]

      X_train = pd.get_dummies(X_train, columns=cat_cols, drop_first=True)
      X_test = pd.get_dummies(X_test, columns=cat_cols, drop_first=True)

      # Align columns so train and test match
      X_train, X_test = X_train.align(X_test, join="left", axis=1, fill_value=0)
```

```
[34]: # Inspect final processed training set
      print(X_train.head())
```

```
          age  trestbps      chol   thalach   oldpeak  sex_1   cp_1   cp_2  \
158  1.421944 -0.973041  5.882908  0.408240  0.527263  False  False   True
199 -0.465841  0.756507 -0.885696 -1.104705 -0.083233   True  False  False
479  0.422528 -0.197726 -0.588175 -0.882213  1.050546   True  False  False
460  1.644036  0.517949  0.118437 -0.214737  0.876119   True  False   True
169 -1.021072 -1.151959 -1.629499 -0.570724 -0.868157  False   True  False
```

Exercises

1. **What are the mean and std of the standardized chol column in the training set? Why aren't they exactly 0 and 1 when checked with pandas.Series.std()?**

Answer : After standardizing the **chol** column, its mean becomes ~0 and std ~1 because StandardScaler uses the population formula (ddof=0). When checked with pandas.Series.std(), it's not exactly 1 since pandas by default uses the sample formula (ddof=1).

2. **Why do we use drop_first=True in one-hot encoding? What problem does it solve?**

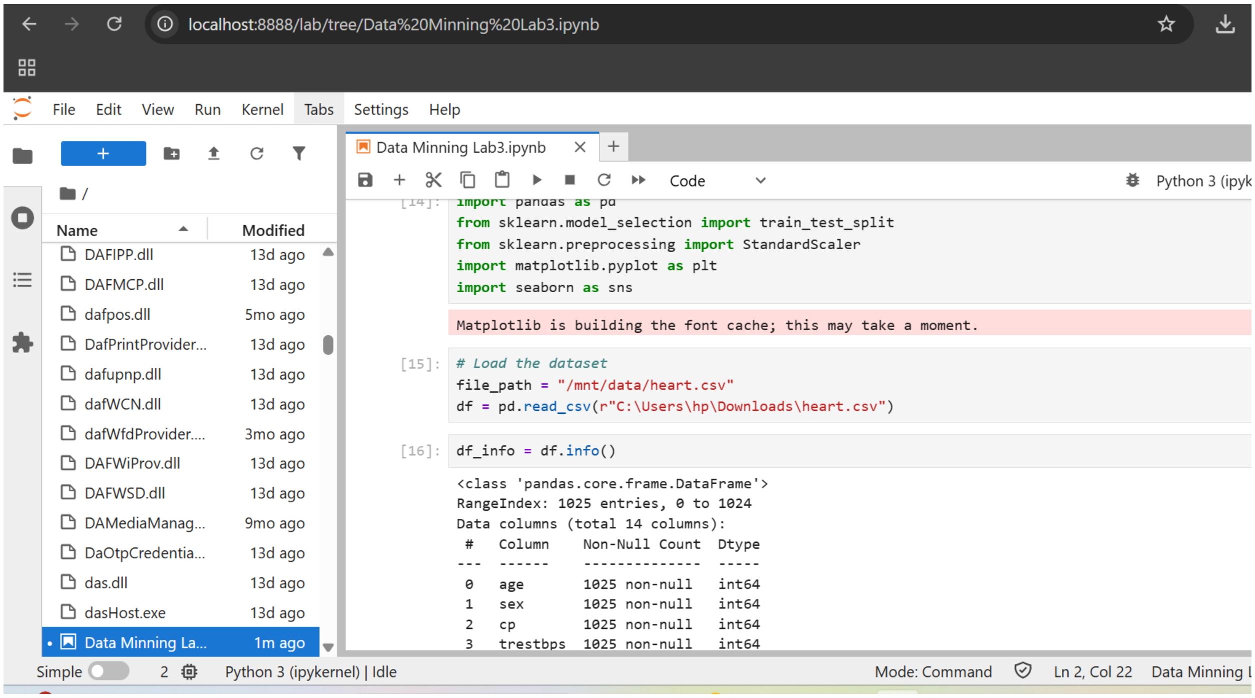
Answer : The drop_first=True option in one-hot encoding avoids the **dummy variable trap** (perfect multicollinearity) by dropping one category and treating it as the baseline.

3. **If a new category of thal appeared in the test set but not in the training set, what would happen with pd.get_dummies? OneHotEncoder(handle_unknown='ignore') in scikit-learn help here?**

Answer : If a new category of **thal** appears in the test set but not in the training set, pd.get_dummies won't create a column for it, leading to mismatched features. Using OneHotEncoder(handle_unknown="ignore") ensures unseen categories are safely ignored, keeping columns consistent.

Data Visualization :

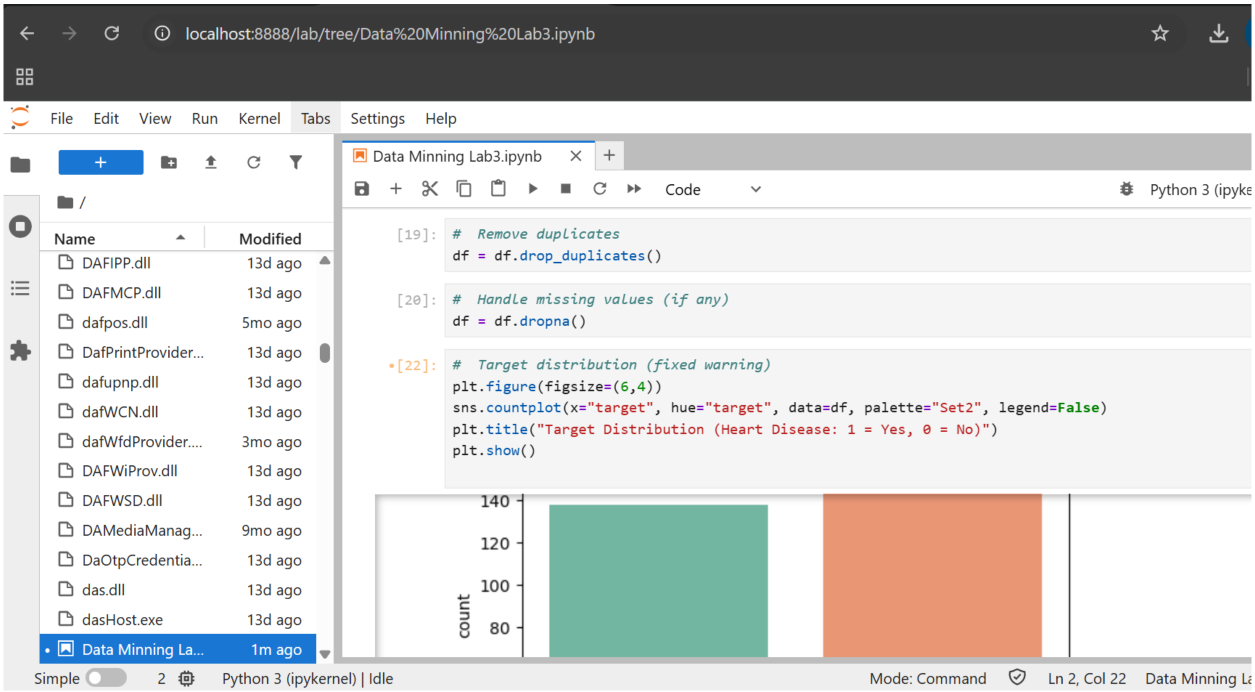
Visualize the heart dataset before doing any cleaning.



```
[14]: import pandas as pd
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      import matplotlib.pyplot as plt
      import seaborn as sns

      Matplotlib is building the font cache; this may take a moment.

[15]: # Load the dataset
      file_path = "/mnt/data/heart.csv"
      df = pd.read_csv(r"C:\Users\hp\Downloads\heart.csv")

[16]: df_info = df.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 1025 entries, 0 to 1024
      Data columns (total 14 columns):
       #   Column    Non-Null Count  Dtype
      ---  ------    --------------  -----
       0   age       1025 non-null   int64
       1   sex       1025 non-null   int64
       2   cp        1025 non-null   int64
       3   trestbps  1025 non-null   int64
```

Here are some useful plots we can make:

1. **Target distribution** (0 = no heart disease, 1 = heart disease)

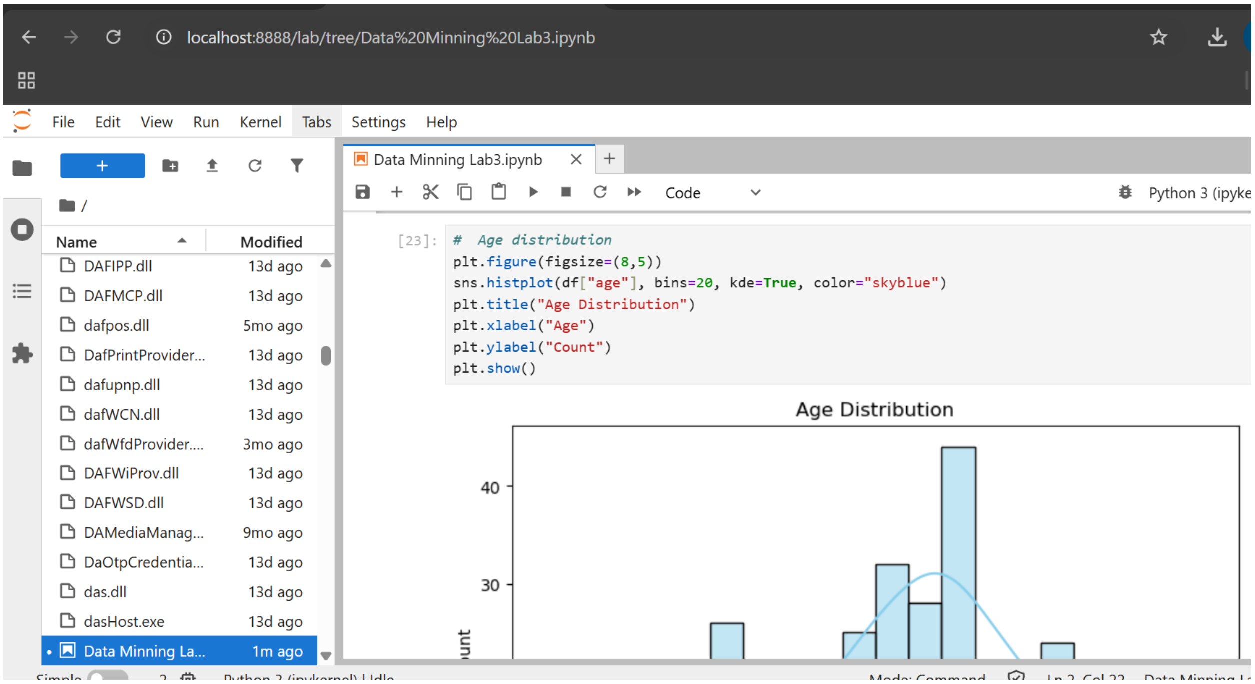


```
[19]: # Remove duplicates
      df = df.drop_duplicates()
```

```
[20]: # Handle missing values (if any)
      df = df.dropna()
```

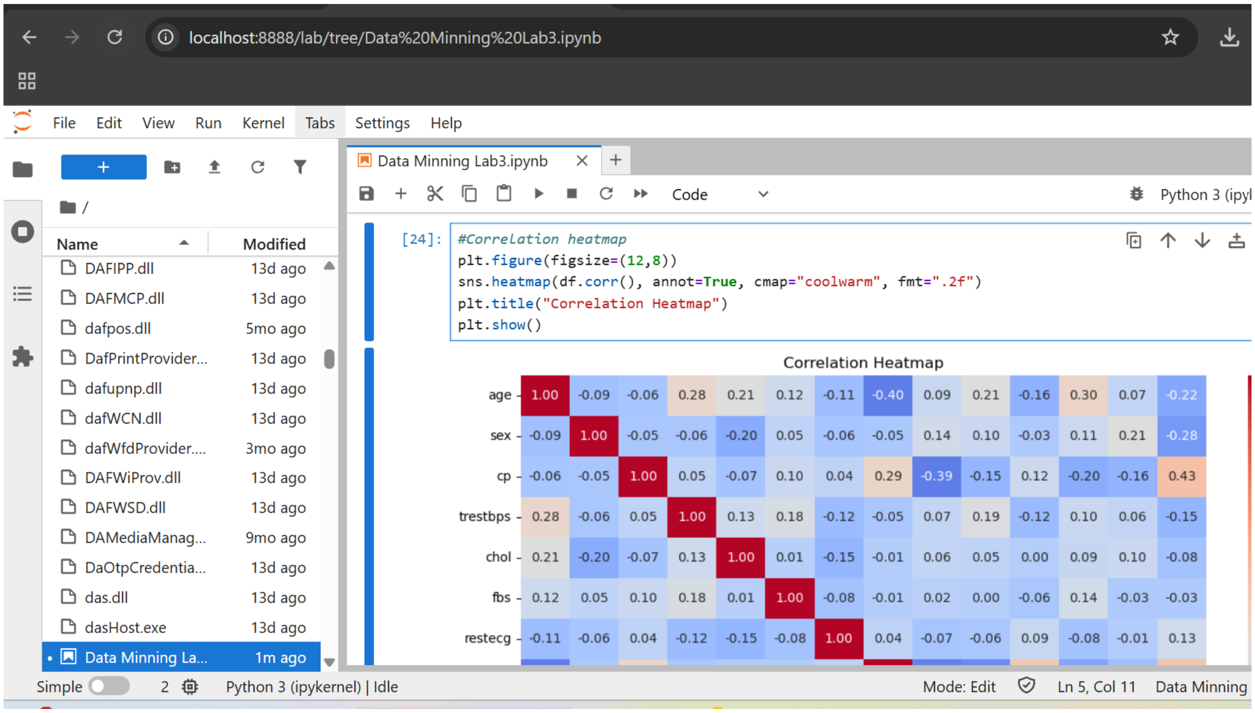
```
[22]: # Target distribution (fixed warning)
      plt.figure(figsize=(6,4))
      sns.countplot(x="target", hue="target", data=df, palette="Set2", legend=False)
      plt.title("Target Distribution (Heart Disease: 1 = Yes, 0 = No)")
      plt.show()
```

2. **Age distribution**



```
[23]: # Age distribution
      plt.figure(figsize=(8,5))
      sns.histplot(df["age"], bins=20, kde=True, color="skyblue")
      plt.title("Age Distribution")
      plt.xlabel("Age")
      plt.ylabel("Count")
      plt.show()
```

3. **Correlation heatmap** of features



Name	Modified
DAFIPP.dll	13d ago
DAFMCP.dll	13d ago
dafpos.dll	5mo ago
DafPrintProvider...	13d ago
dafupnp.dll	13d ago
dafWCN.dll	13d ago
dafWfdProvider....	3mo ago
DAFWiProv.dll	13d ago
DAFWSD.dll	13d ago
DAMediaManag...	9mo ago
DaOtpCredentia...	13d ago
das.dll	13d ago
dasHost.exe	13d ago
Data Minning La...	1m ago

```
[24]: #Correlation heatmap
      plt.figure(figsize=(12,8))
      sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
      plt.title("Correlation Heatmap")
      plt.show()
```

Correlation Heatmap

age	1.00	-0.09	-0.06	0.28	0.21	0.12	-0.11	-0.40	0.09	0.21	-0.16	0.30	0.07	-0.22
sex	-0.09	1.00	-0.05	-0.06	-0.20	0.05	-0.06	-0.05	0.14	0.10	-0.03	0.11	0.21	-0.28
cp	-0.06	-0.05	1.00	0.05	-0.07	0.10	0.04	0.29	-0.39	-0.15	0.12	-0.20	-0.16	0.43
trestbps	0.28	-0.06	0.05	1.00	0.13	0.18	-0.12	-0.05	0.07	0.19	-0.12	0.10	0.06	-0.15
chol	0.21	-0.20	-0.07	0.13	1.00	0.01	-0.15	-0.01	0.06	0.05	0.00	0.09	0.10	-0.08
fbs	0.12	0.05	0.10	0.18	0.01	1.00	-0.08	-0.01	0.02	0.00	-0.06	0.14	-0.03	-0.03
restecg	-0.11	-0.06	0.04	-0.12	-0.15	-0.08	1.00	0.04	-0.07	-0.06	0.09	-0.08	-0.01	0.13