**Step 1: Data Preparation and Exploration**

1. Import necessary libraries like pandas, matplotlib, Seaborn, Plotly, scikit-learn, and transformers.
2. Load the dataset from the provided CSV file into a pandas DataFrame.
3. Data Preprocessing:
   * Remove the 'Patient Id' column as it's not relevant for analysis.
   * Recode the 'Level' column to numerical values, converting 'Medium' to 'High' and 'High' to '1', 'Low' to '0'.
   * Convert the 'Level' column to a numeric data type.

**Step 2: Data Visualization and Exploration**

1. Use Seaborn to create a heatmap to visualize missing data.
2. Create count plots and box plots to explore the relationship between variables like 'Smoking', 'ChestPain', 'Age', and 'Level'.
3. Generate a sorted DataFrame to display the count of smokers by age.
4. Create interactive bar and line charts using Plotly to visualize the number of smokers by age.

**Step 3: Model Building and Evaluation**

1. Split the data into training and testing sets using **train\_test\_split**.
2. Train a Random Forest Classifier model and make predictions.
   * Calculate and display the accuracy score, log loss, and F1 score.
   * Visualize the confusion matrix using a heatmap.
3. Determine the best value for k in the K-Nearest Neighbors (KNN) model.

* Use cross-validation to find the best k.

1. Train a KNN model with the best k and make predictions.

* Calculate and display the accuracy score, log loss, and F1 score.
* Visualize the confusion matrix using a heatmap.

1. Train a K-Means clustering model and predict cluster assignments.

* Calculate and display the accuracy score, log loss, and F1 score.
* Visualize the confusion matrix using a heatmap.

1. Train a Decision Tree Classifier and make predictions.

* Calculate and display the accuracy score, log loss, and F1 score.
* Visualize the confusion matrix using a heatmap.

1. Train a Support Vector Classifier (SVC) model and make predictions.

* Calculate and display the accuracy score, log loss, and F1 score.
* Visualize the confusion matrix using a heatmap.

**Step 4: Zero-Shot Classification (ZSL) for Model Evaluation**

1. Install the 'transformers' library and import necessary components.
2. Use the Zero-shot classification (ZSL) model from the 'transformers' library for text-based predictions.
3. Define candidate labels for ZSL, which in this case are ["High", "Medium", "Low"].
4. Prepare text inputs for ZSL by converting rows from the test dataset to text.
5. Perform ZSL on the test data to predict labels.
6. Extract the predicted labels and evaluate the ZSL predictions:
   * Calculate and display the accuracy score, log loss, and F1 score.
   * Visualize the confusion matrix using a heatmap.

**CODE :**

importimportimportimport pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

importimportimportimporttt cufflinks as cf

import plotly

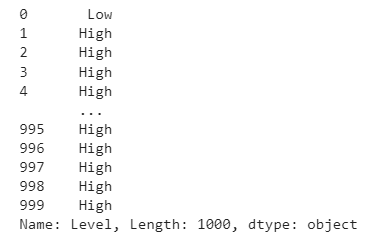
from plotly.offline import init\_notebook\_mode,iplot,plot

importimporttnotebook\_mode(connected=True)

df = pd.read\_csv('/content/Cancer.ccsv')

ccsv'sv'df.drop(['Patient Id'],aaxisxis = 1,inplace=True)

df['Level']

OUTPUT :  


df['Level'].replace('Medium','High',inplace=True)

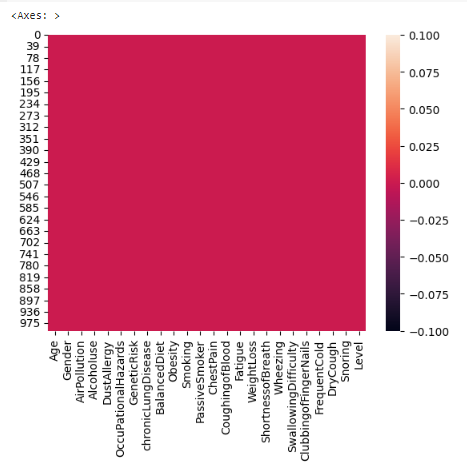
df['Level'].replace('High','1',inplace=True)

df['Level'].replace('Low','0',inplace=True)

df['Level'] = pd.to\_numeric(df['Level'])

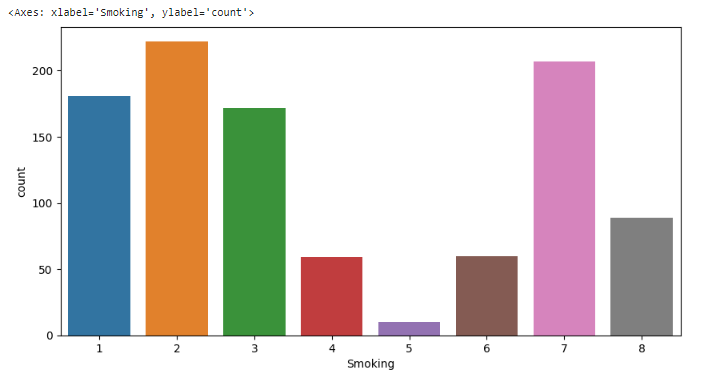
import seaborn as sns

sns.heatmap(df.isnull())

OUTPUT :  


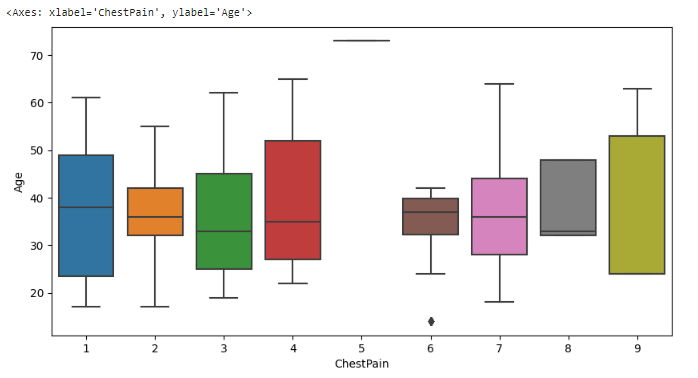
plt.figure(figsize=(10,5))

sns.countplot(x='Smoking',data=df)

OUTPUT :  


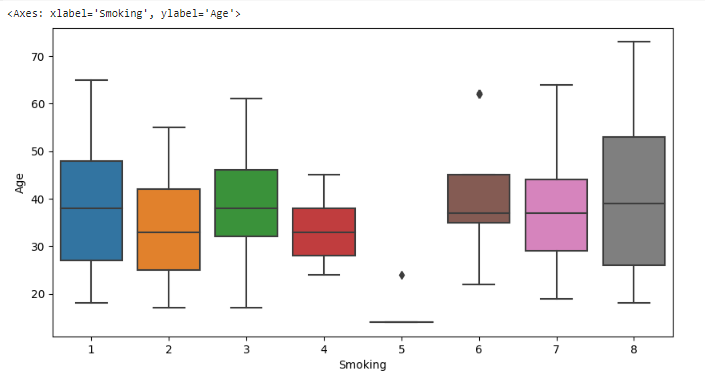
plt.figure(figsize=(10,5))

sns.boxplot(x='ChestPain',y='Age',data = df)

OUTPUT :  


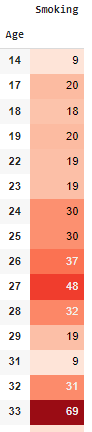
plt.figure(figsize=(10,5))

sns.boxplot(x='Smoking',y='Age',data = df)

OUTPUT :  


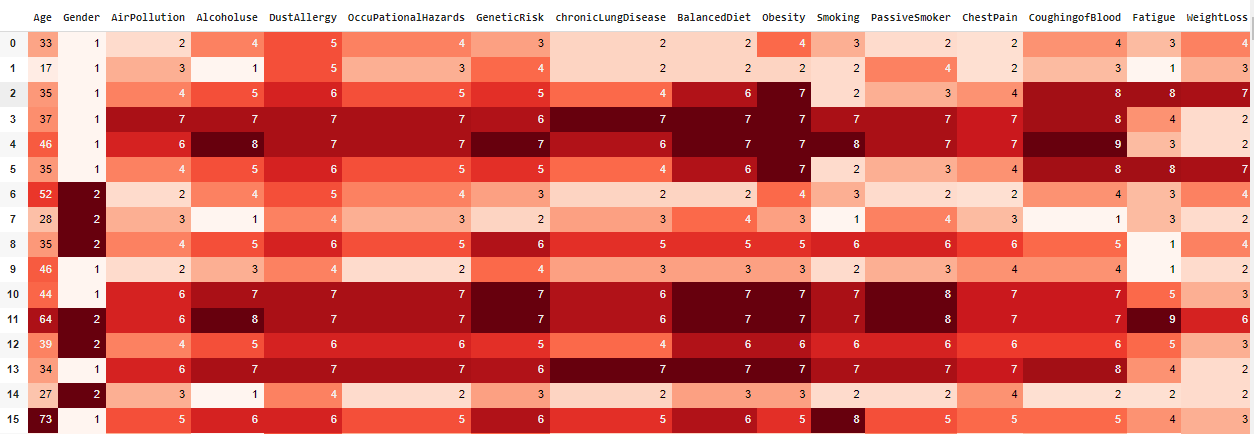
sorted\_smokers = df.groupby('Age')['Smoking'].count().to\_frame()

sorted\_smokers.style.background\_gradient(cmap = 'Reds')

OUTPUT :  


df.style.background\_gradient(cmap = 'Reds')

OUTPUT :



label = df.Age.sort\_values().unique()

target = sorted\_smokers.Smoking

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

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.metrics import log\_loss, f1\_score

from sklearn.model\_selection import cross\_val\_score

import numpy as np

acc\_dict = {}

# create the data

X = df.drop('Level',axis = 1)

y = df['Level']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y)

from sklearn.ensemble import  RandomForestClassifier

# create model

model = RandomForestClassifier()

# fit the data in the model

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

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

print('Accuracy score : ',accuracy\_score(y\_test, y\_pred\_randomF)\*100)

acc\_dict['RFC\_log\_loss'] = log\_loss(y\_test, y\_pred\_randomF)

acc\_dict['RFC\_F!1\_Score'] = f1\_score(y\_test, y\_pred\_randomF,average='weighted')

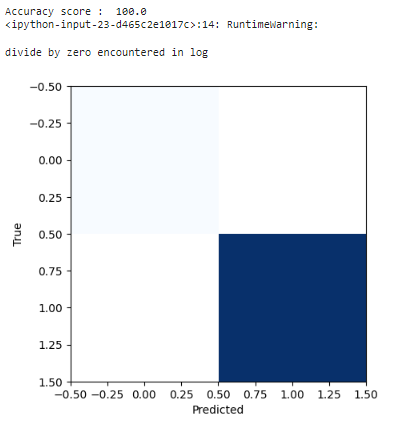
# prediction visualization

plt.imshow(np.log(confusion\_matrix(y\_test,y\_pred\_randomF)),cmap = 'Blues',interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :  


from sklearn.neighbors import KNeighborsClassifier

# to find the best k

score = 0

scores, highscore, bestk = 0, 0, 0

for k in range(3,12):

    knn = KNeighborsClassifier(n\_neighbors=k)

    scores = cross\_val\_score(knn, X\_train, y\_train)

    score = scores.mean()

    if score>highscore:

        highscore = score

        bestk = k

print('Best k is {} with score {}'.format(bestk, highscore))

knn = KNeighborsClassifier(n\_neighbors=bestk)

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

# prediction

y\_predict = knn.predict(X\_test)

print('Accuracy score : ',accuracy\_score(y\_test,y\_predict)\*100)

acc\_dict['KNN\_log\_loss'] = log\_loss(y\_test, y\_predict)

acc\_dict['KNN\_F!1\_Score'] = f1\_score(y\_test, y\_predict,average='weighted')

# prediction visualization

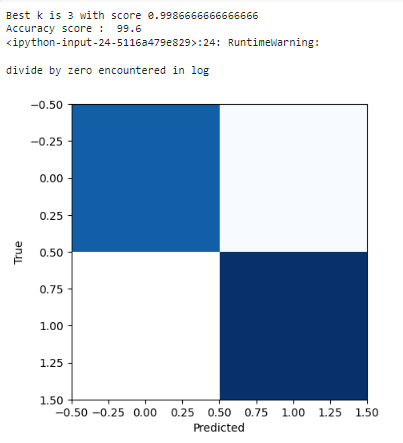
plt.imshow(np.log(confusion\_matrix(y\_test,y\_predict)),cmap = 'Blues',interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :



from sklearn.cluster import KMeans

clf = KMeans()

clf.fit(X\_train)

maxx = clf.predict(X\_test)

print('Accuracy score : ',accuracy\_score(y\_test,maxx)\*100)

acc\_dict['kMeans\_log\_loss'] = log\_loss(y\_test, maxx)

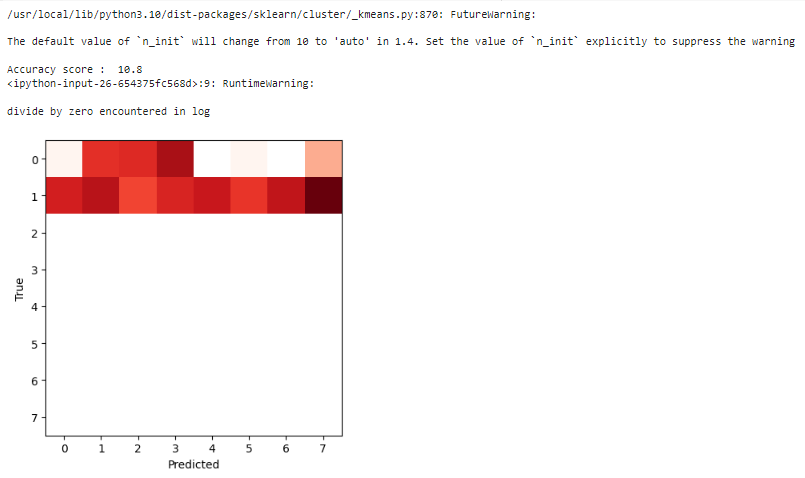
acc\_dict['kMeans\_F1\_Score'] = f1\_score(y\_test, maxx,average='weighted')

plt.imshow(np.log(confusion\_matrix(y\_test,maxx)),cmap='Reds', interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :  


from sklearn.tree import DecisionTreeClassifier

tree\_ = DecisionTreeClassifier()

tree\_.fit(X\_train,y\_train)

y\_pred = tree\_.predict(X\_test)

print('Accuracy score : ',accuracy\_score(y\_test, y\_pred)\*100)

acc\_dict['Tree\_log\_loss'] = log\_loss(y\_test,y\_pred)

acc\_dict['Tree\_f!1\_score'] = f1\_score(y\_test,y\_pred)

# prediction visualization

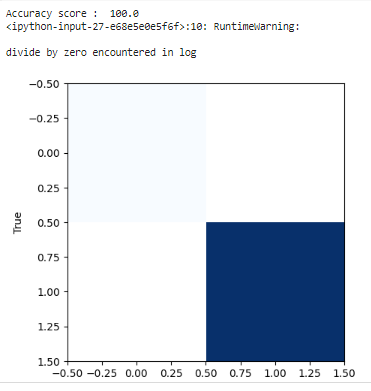
plt.imshow(np.log(confusion\_matrix(y\_test,y\_pred)),cmap = 'Blues',interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :



from sklearn.svm import SVC

model = SVC()

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

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

print('Accuracy score : ',accuracy\_score(y\_test, y\_pred)\*100)

acc\_dict['svc\_log\_loss'] = log\_loss(y\_test,y\_pred)

acc\_dict['svc\_f!1\_score'] = f1\_score(y\_test,y\_pred)

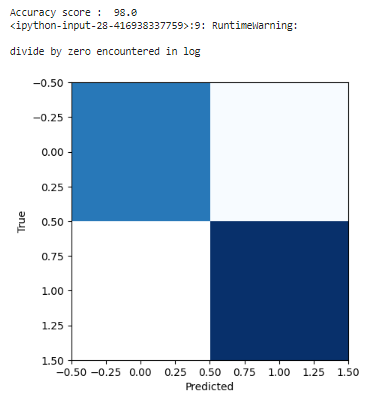
# prediction visualization

plt.imshow(np.log(confusion\_matrix(y\_test,y\_pred)),cmap = 'Blues',interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :  


import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.metrics import accuracy\_score, log\_loss, f1\_score, confusion\_matrix

from transformers import pipeline

# Load the Zero-shot classification model

zsl\_model = pipeline("zero-shot-classification")

# Assuming your dataframe is already loaded as `df`

# Define the candidate labels that the ZSL model can predict

candidate\_labels = ["High", "Medium", "Low"]

X = df.drop('Level', axis=1)

y = df['Level']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)

# Prepare the text inputs for zero-shot classification

X\_test\_text = [" ".join(map(str, row)) for row in X\_test.values]

# Perform zero-shot classification

zsl\_results = zsl\_model(X\_test\_text, candidate\_labels)

# Extract the predicted labels

zsl\_predictions = [result['labels'][0] for result in zsl\_results]

zsl\_predictions = [candidate\_labels.index(label) if label in candidate\_labels else -1 for label in zsl\_predictions]

# Evaluate the zero-shot classification predictions

print('Accuracy score (ZSL): ', accuracy\_score(y\_test, zsl\_predictions) \* 100)

print('Log loss (ZSL): ', log\_loss(y\_test, zsl\_predictions))

print('F1 Score (ZSL): ', f1\_score(y\_test, zsl\_predictions, average='weighted'))

# Visualize the confusion matrix

plt.imshow(np.log(confusion\_matrix(y\_test, zsl\_predictions)), cmap='Blues', interpolation='nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

plt.show()

OUTPUT :

