Sprint-4

```
#ecg classifier
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.set_style('whitegrid')
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
data = pd.read_csv(r'C:\Users\DELL\Desktop\heart
disease/Heart_Disease_Prediction.csv')
data.head()
data.info()
data.describe(include = 'all')
data.isnull().sum()
data.nunique()
data.columns
colm = ['Sex', 'Chest pain type','FBS over 120','EKG results','Exercise
angina','Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart
Disease']
for col in colm:
  sns.countplot(data[col])
  plt.show()
  plt.figure(figsize=(12,10))
corr = data.corr()
sns.heatmap(corr, annot = True, linewidths= 0.2, linecolor= 'black', cmap =
'afmhot')
data.columns
X = data[['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over
120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium']]
y = data['Heart Disease']
print(X.shape,y.shape)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
random_state=42529)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
train_convert = {"Absence":0,"Presence":1}
y_train = y_train.replace(train_convert)
test_convert = {"Absence":0,"Presence":1}
y_test = y_test.replace(test_convert)
mms = MinMaxScaler()
X_train = mms.fit_transform(X_train)
X_test = mms.fit_transform(X_test)
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
pred = rf.predict(X_test)
cm = confusion_matrix(y_test,pred)
print(classification_report(y_test,pred))
```

```
sns.heatmap(cm, annot = True, fmt = 'g', cbar = False, cmap = 'icefire',
linewidths= 0.5, linecolor= 'grey')
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
print("Accuracy Score = {}".format(round(accuracy_score(y_test,pred),5)))
#multiple classifier


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
df = pd.read_csv(r'C:\Users\DELL\Desktop\heart
disease/Heart_Disease_Prediction.csv')
df.dtypes
df.head()
df.isnull().sum()
format(len(df[df.duplicated()]))
name = df.columns
num_var = ['Age', 'BP', 'Cholesterol', 'Max HR', 'Heart Disease']
cat_var = [item for item in name if item not in num_var]

num_var_data = df[df.columns & num_var]
num_var_data.describe()
num_var_data.corr()
sns.heatmap(num_var_data.corr(), cmap="YlGnBu", annot=True)
sns.pairplot(num_var_data)
num_var_data[num_var_data['Cholesterol'] > 500]
sns.pairplot(num_var_data, hue = 'Heart Disease')

x = df.drop(['Heart Disease'], axis = 1)
y = df['Heart Disease']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
model = LogisticRegression()
model.fit(X_train, y_train)
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
from sklearn.preprocessing import LabelEncoder
x_train_enc = X_train

le = LabelEncoder()
le.fit(y_train)
y_train_enc = le.transform(y_train)


from sklearn.inspection import permutation_importance
```

```
model.fit(x_train_enc, y_train_enc)

results = permutation_importance(model, x_train_enc, y_train_enc,
scoring='neg_mean_squared_error')


importance = results.importances_mean


for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
    # plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
df.columns
selected_feature = ['Sex','Max HR', 'Number of vessels fluro', 'Thallium']
print(selected_feature)
data = df[df.columns & selected_feature]

X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.33)
model = LogisticRegression()
model.fit(X_train, y_train)
r_sq = model.score(data, y)
print(f"coefficient of determination: {r_sq}")

models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

results = []
names = []
scoring = 'accuracy'

for name, model in models:
    kfold = KFold(n_splits=10, random_state=7, shuffle = True)
    cv_results = cross_val_score(model, data, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
#main code
# -*- coding: utf-8 -*-

import numpy as np
import pickle
from flask import Flask, request, render_template
```

```
# Load ML model
model = pickle.load(open('model.pkl', 'rb'))

# Create application
app = Flask(__name__)

# Bind home function to URL
@app.route('/')
def home():
    return render_template('Heart Disease Classifier.html')

# Bind predict function to URL
@app.route('/predict', methods =['POST'])
def predict():

    # Put all form entries values in a list
    features = [float(i) for i in request.form.values()]
    # Convert features to array
    array_features = [np.array(features)]
    # Predict features
    prediction = model.predict(array_features)

    output = prediction

    # Check the output values and retrive the result with html tag based on
the value
    if output == 1:
        return render_template('Heart Disease Classifier.html',
                               result = 'The patient is not likely to have
heart disease!')
    else:
        return render_template('Heart Disease Classifier.html',
                               result = 'The patient is likely to have heart
disease!')

if __name__ == '__main__':
#Run the application
    app.run()

    #model
# -*- coding: utf-8 -*-
"""
Created on Fri Nov 18 12:39:46 2022

@author: DELL
"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as py
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pickle
```

```
df=pd.read_csv("Heart_Disease_Prediction.csv")
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values


std=StandardScaler()
x=std.fit_transform(x)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

model=RandomForestClassifier()
model.fit(x_train,y_train)
predictions=model.predict(x_test)
accuracy = accuracy_score(y_test,predictions)

def predict_heart_disease(parameter_list):
    return model.predict(parameter_list)[0]

pickle.dump(model, open('model.pkl','wb'))
```