import numpy as np import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import matplotlib.pyplot as plt import seaborn as sns

from datetime import datetime import sklearn

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LogisticRegression

noshow\_appointments = pd.read\_csv("KaggleV2-May-2016.csv") (noshow\_appointments.head())

df.AppointmentDay = df.AppointmentDay.apply(np.datetime64) df['WeekDay'] = df['AppointmentDay'].dt.day df.ScheduledDay = df.ScheduledDay.apply(np.datetime64) df['DayScheduled'] = df['ScheduledDay'].dt.day

df.drop(['PatientId', 'AppointmentID'], axis=1, inplace=True)

# Converting 'Gender' and 'Noshow' from object format to integer format. df.Gender = df.Gender.apply(lambda x: 1 if x == 'M' else 0) df['Noshow'] = df['Noshow'].replace('Yes',1)

df['Noshow'] = df['Noshow'].replace('No',0) range\_df = pd.DataFrame()

range\_df['Age'] = range(95)

men = range\_df.Age.apply(lambda x:len(df[(df.Age == x) & (df.Gender == 1)])) women = range\_df.Age.apply(lambda x:len(df[(df.Age == x) & (df.Gender == 0)]))

plt.plot(range(95), men, color = 'b') plt.plot(range(95), women, color = 'g') plt.legend([1,0])

plt.xlabel('Age') plt.ylabel('Frequency') plt.title('Gender based difference')

men\_Hypertension = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 1) & (df.Hypertension == 1)])) women\_Hypertension = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 0) & (df.Hypertension == 1)]))

men\_Diabetes = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age

== x) & (df.Gender == 1) & (df.Diabetes == 1)])) women\_Diabetes = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 0) & (df.Diabetes == 1)]))

men\_Alcoholism = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 1) & (df.Alcoholism == 1)])) women\_Alcoholism = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 0) & (df.Alcoholism == 1)]))

men\_Handicap = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age

== x) & (df.Gender == 1) & (df.Handicap == 1)])) women\_Handicap = range\_df[range\_df.columns[0]].apply(lambda x: len(df[(df.Age == x) & (df.Gender == 0) & (df.Handicap == 1)])) plt.figure(figsize = (10,10))

plt.subplot(2,2,1) plt.plot(range(95),men\_Hypertension/men) plt.plot(range(95),women\_Hypertension/women, color = 'r') plt.title('Hypertension')

plt.legend([1,0], loc = 2) plt.xlabel('Age') plt.ylabel('Frequency')

plt.subplot(2,2,2) plt.plot(range(95),men\_Diabetes/men) plt.plot(range(95),women\_Diabetes/women, color = 'r') plt.title('Diabetes')

plt.legend([1,0], loc = 2) plt.xlabel('Age') plt.ylabel('Frequency')

plt.subplot(2,2,3) plt.plot(range(95),men\_Alcoholism/men) plt.plot(range(95),women\_Alcoholism/women, color = 'r') plt.title('Alcoholism')

plt.legend([1,0], loc = 2) plt.xlabel('Age') plt.ylabel('Frequency')

plt.subplot(2,2,4) plt.plot(range(95),men\_Handicap/men) plt.plot(range(95),women\_Handicap/women, color = 'r') plt.title('Handicap')

plt.legend([1,0], loc = 2) plt.xlabel('Age') plt.ylabel('Frequency')

x = df.drop(['Noshow','Neighbourhood','ScheduledDay','AppointmentDay'], axis=1)

y = df['Noshow']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.4, random\_state=42)

logreg = LogisticRegression() logreg.fit(x\_train, y\_train) y\_pred = logreg.predict(x\_test)

acc\_log = round(logreg.score(x\_train, y\_train) \* 100, 2) acc\_log

from sklearn.neighbors import KNeighborsClassifier from sklearn.tree import DecisionTreeClassifier from sklearn.metrics import accuracy\_score

from sklearn.svm import SVC

# KNN Algorithm

knn = KNeighborsClassifier() knn.fit(x\_train, y\_train) y\_pred\_knn = knn.predict(x\_test)

acc\_knn = round(accuracy\_score(y\_test, y\_pred\_knn) \* 100, 2)

# Decision Tree Algorithm

decision\_tree = DecisionTreeClassifier() decision\_tree.fit(x\_train, y\_train) y\_pred\_dt = decision\_tree.predict(x\_test)

acc\_dt = round(accuracy\_score(y\_test, y\_pred\_dt) \* 100, 2)

print("Logistic Regression Accuracy:", acc\_log) print("KNN Accuracy:", acc\_knn) print("Decision Tree Accuracy:", acc\_dt)

svm = SVC() svm.fit(x\_train, y\_train)

y\_pred\_svm = svm.predict(x\_test)

acc\_svm = round(accuracy\_score(y\_test, y\_pred\_svm) \* 100, 2)

# Print SVM Accuracy print("SVM Accuracy:", acc\_svm)

from sklearn.metrics import classification\_report

# Logistic Regression Classification Report print("Logistic Regression Classification Report:") print(classification\_report(y\_test, y\_pred))

# KNN Classification Report print("\nKNN Classification Report:")

print(classification\_report(y\_test, y\_pred\_knn))

# Decision Tree Classification Report print("\nDecision Tree Classification Report:") print(classification\_report(y\_test, y\_pred\_dt)) print("\nSVM Classification Report:") print(classification\_report(y\_test, y\_pred\_svm))

32

from sklearn.metrics import precision\_recall\_fscore\_support

# Logistic Regression Precision, Recall, F1-score

precision\_lr, recall\_lr, f1\_lr, \_ = precision\_recall\_fscore\_support(y\_test, y\_pred, average='weighted')

print("Logistic Regression:") print("Precision:", precision\_lr) print("Recall:", recall\_lr) print("F1-score:", f1\_lr)

# KNN Precision, Recall, F1-score precision\_knn, recall\_knn, f1\_knn, \_ =

precision\_recall\_fscore\_support(y\_test, y\_pred\_knn, average='weighted') print("\nKNN:")

print("Precision:", precision\_knn) print("Recall:", recall\_knn) print("F1-score:", f1\_knn)

# Decision Tree Precision, Recall, F1-score

precision\_dt, recall\_dt, f1\_dt, \_ = precision\_recall\_fscore\_support(y\_test, y\_pred\_dt, average='weighted')

print("\nDecision Tree:") print("Precision:", precision\_dt) print("Recall:", recall\_dt) print("F1-score:", f1\_dt)

# SVM Precision, Recall, F1-score precision\_svm, recall\_svm, f1\_svm, \_ =

precision\_recall\_fscore\_support(y\_test, y\_pred\_svm, average='weighted') print("\nSVM:")

print("Precision:", precision\_svm) print("Recall:", recall\_svm) print("F1-score:", f1\_svm)

from sklearn.metrics import confusion\_matrix

# Confusion matrix for KNN

cm\_knn = confusion\_matrix(y\_test, y\_pred\_knn) print("Confusion Matrix for KNN:\n", cm\_knn)

# Confusion matrix for Decision Tree

cm\_dt = confusion\_matrix(y\_test, y\_pred\_dt) print("\nConfusion Matrix for Decision Tree:\n", cm\_dt) from sklearn.metrics import confusion\_matrix

# SVM Algorithm svm = SVC()

svm.fit(x\_train, y\_train)

y\_pred\_svm = svm.predict(x\_test) 33

acc\_svm = round(accuracy\_score(y\_test, y\_pred\_svm) \* 100, 2) # Print SVM Accuracy

print("SVM Accuracy:", acc\_svm)

# Confusion matrix for SVM

cm\_svm = confusion\_matrix(y\_test, y\_pred\_svm) print("\nConfusion Matrix for SVM:\n", cm\_svm) from sklearn.metrics import confusion\_matrix

# Compute confusion matrix for Logistic Regression conf\_matrix\_lr = confusion\_matrix(y\_test, y\_pred) print("\nConfusion Matrix for Logistic Regression:")

print(conf\_matrix\_lr)