```
#%%
import pandas as pd


import matplotlib.pyplot as plt

import seaborn as sns


import warnings
warnings.filterwarnings("ignore")

np.set_printoptions(suppress = True)


from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics.pairwise import
euclidean_distances
from sklearn.cluster import DBSCAN

from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

# import functions from scipy to perform clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cophenet

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis

from surprise import KNNWithMeans,SVDpp
```

```
from surprise import accuracy
#%% md
### Section B 1
#%%
df_data  = pd.read_csv("diabetes.csv")
#%%
df_data.head(10)
#%%
df_data.shape
#%%
df_data.dtypes
#%%
df_data.describe()
#%%
df_data.isnull().sum()
#%%
df_data.Glucose = df_data.Glucose.replace(0,
df_data['Glucose'].mean())
#%%
def replace_q(s):
    if s == '20c':
        return 500.0
    elif s=='!':
        return 0.0
    else:
        return float(s)
#%%
gmean = df_data.Glucose.mean()
df_data['Glucose'] = df_data['Glucose'].apply(
lambda x : gmean if x == 0 else float(x))
#%%
df_data.BloodPressure = df_data.BloodPressure.
replace(0,df_data['BloodPressure'].mean())
df_data.SkinThickness = df_data.SkinThickness.
replace(0, df_data['SkinThickness'].mean())
df_data.Insulin = df_data.Insulin.replace(0,df_data
['Insulin'].mean())
df_data.BMI = df_data.BMI.replace(0,df_data['BMI'].
mean())
#%%
df_data.describe()
```

```
#%%
plt.rcParams['figure.figsize'] = [15,8]
df_data.boxplot()

plt.show()
#%%
Q1 = df_data.quantile(0.25)
Q3 = df_data.quantile(0.75)


IQR = Q3 - Q1

df_data = df_data[~((df_data <(Q1 - 1.5*IQR))|(
   df_data >(Q3 + 1.5*IQR))).any(axis=1)]

df_data.shape
#%%
plt.rcParams['figure.figsize'] = [15,8]
df_data.boxplot()

plt.show()
#%%
X = df_data.drop('Outcome' , axis=1)

y = df_data['Outcome']
#%%
X_scle = StandardScaler().fit_transform(X)
#%% md
 90% variance


#%%
cov_matrix = np.cov(X_scle.T)
eig_vals,eig_vecs = np.linalg.eig(cov_matrix)

print("Eigen Vectors \n%s",eig_vecs)
print("Eigen Values \n%s",eig_vals)
#%%
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals,
   reverse=True)]
```

```
cum_var_exp = np.cumsum(var_exp)
print("Cumulative variance explained",cum_var_exp)
#%%
plt.plot(cum_var_exp)
plt.axvline(x=5, ymax=90, ls='--', color='r')
plt.axhline(y=90, xmax=90, ls='--', color='r')
#%%
print("Top 5 Eigen Vectors are :",sorted(eig_vals,
    reverse=True)[0:5])
#%%
var_exp
#%%
pca = PCA(0.90)
X_PCA = pca.fit(X_scle)
#%%
X_PCA.explained_variance_
#%%
X_PCA.components_
#%%
X_PCA.explained_variance_[:5]
#%%
X_PCA.components_[:5]
#%% md
### Section B 2
#%%
# 95% variance
pca1 = PCA(0.95)
X_PCA_1 = pca1.fit_transform(X)
#%%
from yellowbrick.cluster import
    SilhouetteVisualizer
n_clusters=[2,3,4,5,6,7,8,9,10]

for K in n_clusters:
    model=KMeans(n_clusters=K,random_state=42)
    viz = SilhouetteVisualizer(model).fit(X_PCA_1)
    viz.show()
#%%
km = KMeans(n_clusters=2,random_state=42)
model = km.fit(X_PCA_1)
df_data['Cluster'] = model.labels_
```

```
#%%
df_data.head()
#%%
df_data[df_data['Cluster']==0].describe()
#%%
df_data[df_data['Cluster']==1].describe()
#%% md
**Interpretation:** The above summary shows that the average Insulin in this cluster is 78 +/- 15 dev. On average, the gl level is 113.86.
#%% md
# Optimal number of clusters  is 2
#%% md
# Section B 3
#%%
link = linkage(X_PCA_1, method='single')
dendrogram(link)
plt.show()
#%%
link = linkage(X_PCA_1, method='centroid')
dendrogram(link)
plt.show()
#%%
link = linkage(X_PCA_1, method='ward')
dendrogram(link)
plt.show()
#%%
link = linkage(X_PCA_1, method='complete')
dendrogram(link)
plt.show()
#%% md
# Section B 4
#%%
model_Km = KMeans(n_clusters = 4, random_state=40)
model_Km.fit(X_PCA_1)

df_data['Cluster_KM'] = model_Km.labels_
#%% md
model = KMeans(n_clusters=4)
distances = model.fit_transform(X_PCA_1)
variance = 0
```

```
i = 0
for label in model.labels_:
    variance = variance + distances[i][label]
    i = i + 1
model.labels_
#%%
#Bar plot
#%% md
# Section C 1
#%% md
# Normal
#%%
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.20,
    random_state=40 )

rf = RandomForestClassifier(n_estimators=15)
model = rf.fit(X_train,y_train)
y_pred = model.predict(X_test)

print(accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
cm
#%% md
# PCA
#%%
pca = PCA(0.90 )
X_PCA_2 = pca.fit_transform(X)
#%%
X_train, X_test, y_train, y_test =
    train_test_split(X_PCA_2, y, test_size=0.20,
    random_state=40 )

rf = RandomForestClassifier(n_estimators=20)
model = rf.fit(X_train,y_train)
y_pred = model.predict(X_test)

print(accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
cm
#%% md
```

```
229 # SVD
230 #%%
231 svd = TruncatedSVD(8)
232 X_SVD = svd.fit_transform(X)
233 #%%
234 np.cumsum(svd.explained_variance_/sum(svd.
    explained_variance_)*100)
235 #%%
236 svd = TruncatedSVD(3)
237 X_SVD = svd.fit_transform(X)
238 #%%
239 X_train, X_test, y_train, y_test =
    train_test_split(X_SVD, y, test_size=0.20,
    random_state=40 )
240
241 rf = RandomForestClassifier(n_estimators=20)
242 model = rf.fit(X_train,y_train)
243 y_pred = model.predict(X_test)
244
245 print(accuracy_score(y_test, y_pred))
246 cm = confusion_matrix(y_test, y_pred)
247 cm
248 #%% md
249 # LDA
250 #%%
251 #num of classes (2) ==> n_classes - 1 = 1
252 lda = LinearDiscriminantAnalysis(n_components = 1)
253 X_lda = lda.fit_transform(X, y)
254 #%%
255 X_train, X_test, y_train, y_test =
    train_test_split(X_lda, y, test_size=0.20,
    random_state=40 )
256
257 rf = RandomForestClassifier(n_estimators=20)
258 model = rf.fit(X_train,y_train)
259 y_pred = model.predict(X_test)
260
261 print(accuracy_score(y_test, y_pred))
262 cm = confusion_matrix(y_test, y_pred)
263 cm
264 #%%
```

```

#%% md
# Scetion C 2
#%%
df_rating = pd.read_csv('ratings.csv')
df_rating.head()
#%%
df_rating.drop('timestamp', axis=1, inplace=True)
df_rating.head()
#%%
df_rating[['movieId', 'rating']].groupby('movieId'
    ).mean().sort_values(by='rating', ascending=False)
#%%
popularity_tabel = df_rating.groupby('movieId').
    agg({'rating':'mean', 'userId':'count'})
popularity_tabel.rename({'userId':'TotalCount'},
    axis=1, inplace=True)
popularity_tabel = popularity_tabel.sort_values('
    rating', ascending= False)
popularity_tabel.head()
#%%
 popularity_tabel[popularity_tabel['TotalCount'
    ] > 250].head(5)
#%% md
# Scetion C 3
#%%
df_rating
#%%
from surprise import Dataset,Reader
from surprise.model_selection import
    train_test_split,cross_validate
from surprise import KNNWithMeans,SVDpp
from surprise import accuracy
#%%
reader = Reader(rating_scale=(1,5))
rating_data = Dataset.load_from_df(df_rating[['
    userId','movieId','rating']],reader)

[trainset,testset] = train_test_split(rating_data,
    test_size=0.15,shuffle=True)
trainsetfull=rating_data.build_full_trainset()
```

```
298 #%%
299 print("Number of users :",trainsetfull.n_users,'\n
    ')
300 print("Number of movies :",trainsetfull.n_items,'\
    n')
301 #%%
302 alg=SVDpp()
303 alg.fit(trainsetfull)
304 #%%
305 alg.predict(uid=710,iid=166534)
306 #%%
307 results=cross_validate(algo=alg,data=rating_data,
    measures=['rmse'],cv=5,return_train_measures=True)
308 #%%
309
```