School of Public Health
College of Health Sciences,

Masters in Health Informatics

MHI8101: Health Analytics

Facilitator: MRs. Irene Wanyana Ssali (MakSPH)

Submission by
Mark Outeke
2100703309
2021/HD07/3309U

Contents

Background

Machine learning approaches in the decision-making processes of public health management are taking shape. Predicting outcomes through reproducible statistical approaches is evolving in a manner that saves time while ensuring that they are reliable. Tuberculosis – as public health problem globally – requires a consensus of approaches to managing several outcome such as MDRTB. These approaches combined with machine leaning techniques can deliver effective evidence for policy change in clinical workflows that can increase treatment success for tuberculosis.

This submission exploits a few of the machine learning techniques on a data set of a three-year prospective cohort household follow-up study in South Lima and Callao, Peru, which is low middle-income country.

The data set from the study.

This dataset – from a study on the incidence of TB disease among household contacts – comprises 4117 unique individuals across 700 households. The cohorts of contacts from two index patient groups: MDRTB, 213; Drug susceptible TB, 487; were followed up for three years(Grandjean et al., 2015). Machine learning requires a larger enough dataset and this fits the criteria. It is a mixed variable dataset with both categorical and continuous variables, which makes it interesting when dealing with cleaning and wrangling processes.

Key Variables include:

- Family Code
- Individual Code
- Incident TB Disease
- MDR or Sensitive Household
- Follow Up Time
- Index Sex
- Index Education
- Index Sputum Smear Grade
- Index Diabetes
- Index Incarceration
- Index Hospitalization
- Index Alcohol Use
- Index Tobacco Use
- Index Side Effects of Medication
- Index Cough Duration (days)
- Index Work
- MDR Sensitive Numeric
- Socio Economic Tertile
- Contact Age
- Contact Sex
- Contact Chemotherapy
- Contact Work
- Contact HIV
- Index Strain Genotype
- Contact Diabetes
- Contact-Index Roomshare
- Contact Previous TB History
- Index History of TB Before
- Index HIV Status

The variables collected among the index patients and their contacts inspired the prediction analysis of this ML assignment.

The intention therefore, is to explore supervised and unsupervised machine learning techniques that can correctly predict the incidence of TB (Outcome) among contacts of the index Tb and MDRTB cases. I deployed known statistical methods in understanding the size of this dataset, data types and scale used in measuring and recording the observations. The format of the dataset is in need of preprocessing and this was done both manually and technically depending of the column and features.

Methods used.

This work was done in python (Python 3.9.12) and Anaconda3 with Jupyter notebooks (server 6.4.8) as a choice for Machine learning development Environment coding preferred by data scientist globally. A summary of the methods is tabulated below.

Literature Review	
	Study paper – Article cited from the dataset got from Plos One.
	Towards Data science
	Skitlearn community
	Kagggle, google datasets, IBM datasets
Environment.	**Details**
	Anaconda3
	Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
	Jupyter Notebook,
	Pandas, Numpy, Seaborn, sklearn, matplotlib, IPython- Display.
Activities	**Details**
Wrangling *Preprocessing* *Imputation* *Visualizations*	Loading and Investigating the data. • Checking for data types of features with • Identifying null values • Identifying duplicates • Summarizing counts of column observation • Manual investigation of every column • Categorizing of data • Dropping duplicates • Imputations through lambda methods of fill() in python • Boxplots for investigating distribution of continuous variables • Bar charts for categorical distribution groups • Heat maps for assign missing values • Decision trees from Random forest • Confusion Matrixes • Accuracy levels of each method in the supervised learning section • ROC curves. • Feature performance bar plot • Correlation map •

Key Continuous variables	1. Follow up time 2. Cough duration (days) 3. Contact Age
Unsupervised	**Details**
Kmeans Clustering	Kmeans algorithm used to determine distribution of continuous variables
PCA Clustering	Agglomerative Clustering with PCA 3 components visualized
Kmeans with PCA scores Clustering	Combining the 3 PCA components to Kmeans using pca scores to experiment fitting, resulted into over fitting when checked with logistic regression.
Data Splitting	Manually and through code
Distribution plotting	Plots by matplotlib and seaborn to plot clustering with scatter plots, regplots for regression investigation and dendograms for classes. Confusion plots and ROC curves of the regression tests.
Supervised	**Details**
Logistic regression	Since data is mostly categorical, and outcome which is binary, one of the best ML techniques is logistic regression. This method scored higher at 100%, but investigations can suggest otherwise. This prompted the application of other methods such SVM, Random tree classification an
Support Vector Machine	Applied with sigmoid kernel because the outcome of interest is a binary. A prediction of 96% was recorded for the train and test data.
Random forest Classification	This had an accuracy of 92% with the same dataset as the above techniques. Trees of 20 classes were provided for the 15 features passed. The intention was to keep the same number of high dimensionality to signify the importance and stress the algorithm. Further investigations can be carried out for dimensions below (5-10).
Cross validation of 10 folds	Cross validation done for Random forest classification intentionally to demonstrate its effect with 10 folds, which affected the performance of the algorithm.
Feature selection	Two methods applied for feature selection. ***Chi-square*** was applied to obtain the ***p-values*** for all the independent variables and were sorted. Out of which ***15 were selected*** for the training set.

Presentation of Results.

In this section, I will display results from the various methods used for the tabulated techniques. Comprised of visualizations and tables accompanied by snippets of outputs, this section will give evidence for the performance of the methods and techniques applied.

The Unsupervised Learning.

The role of this section was to investigate the data and its distribution. The section of work starts with the distribution of clusters viewed with a selected number of visualizations. For instance, to understand the distribution of clusters of contacts who developed TB is visualized as a dendograms below.

Figure 1 Dendograms for the distribution cluster of families and the incident of TB

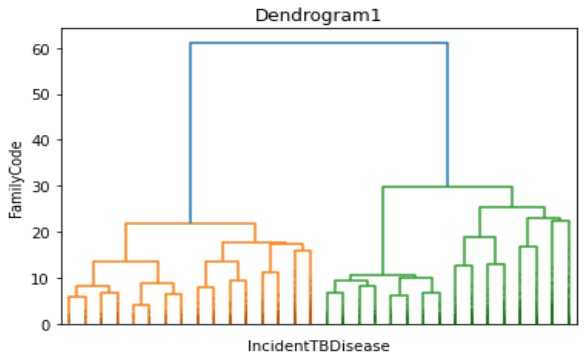


Figure 2 Distribution of Incident of TB with the Two categories of MDR and SENS TB

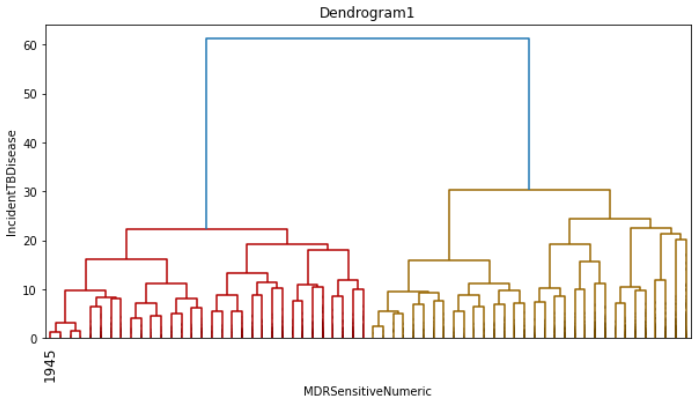
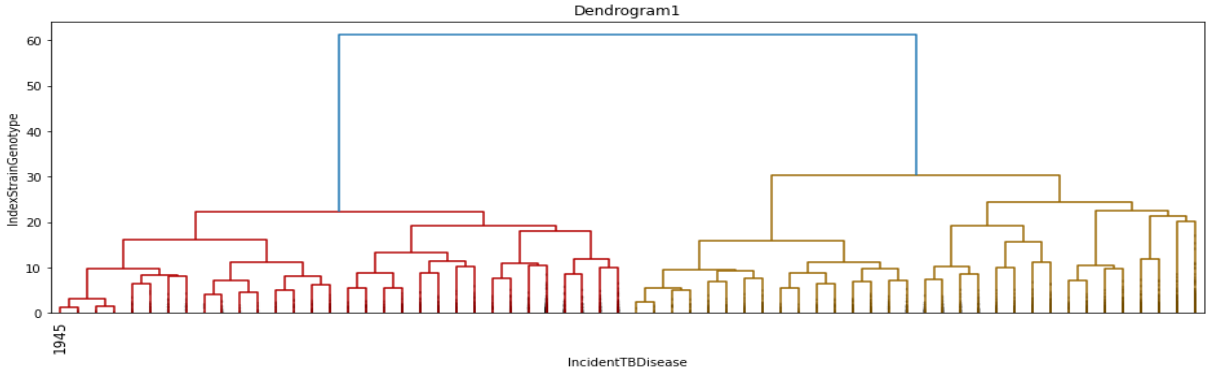
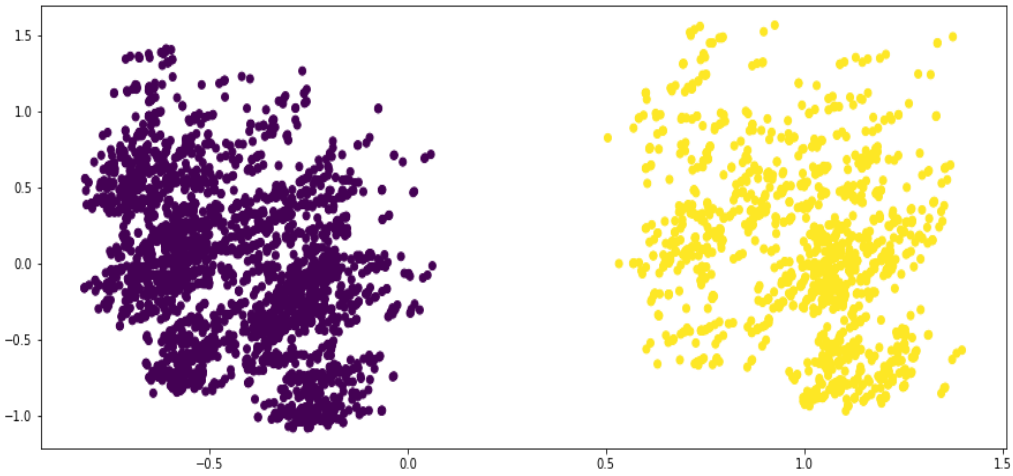


Figure 3 This shows the distribution of cluster with the different Strain Genotype



Another Visualization is for the Cluster of the two clusters generated with normalized data for the Agglomerative clustering method as mentioned in the methods above. This was to reduce the components to three that are later passed onto the kMeans methods with PCA score.

Figure 4 Agglomerative clustering with PCA demonstarion to reduce the dimentionality to only 3 compnents



The next results demonstrate the how the number of best clusters was got through kmeans. It is important to remember that this is experimental for a binary outcome and the purpose for which is to reduce dimensions, as we understand the data. The evidence is taken from the visualization below with a kink at the position 2 of the clusters axis. The other is a formatted kmeans with PCA clustering. The following section will show results of the investigation of the applied 2 clusters with a logistic regression model where if fitted with the normalized data performs to only 50% - not a good model at the moment. The Second plot of kmeans attempts to be fluid, almost signifying a non-linear (continuous behavior of observations after fitting with PCA scores).

Figure 5 Kink at the Elbow for selection of 2 Clusters

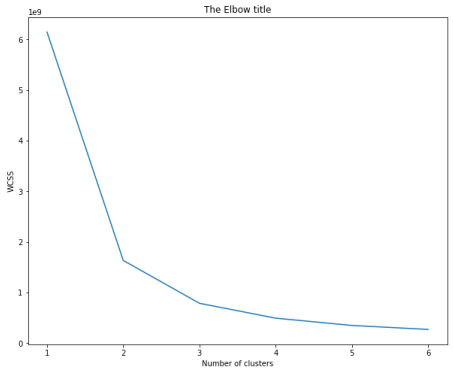
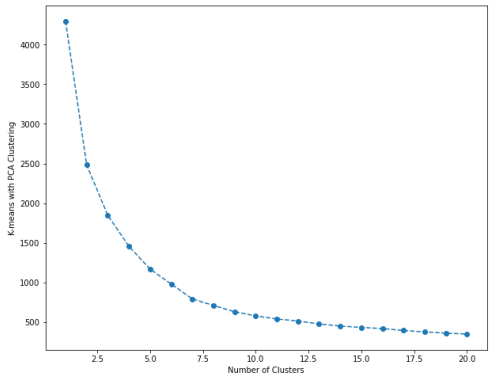


Figure 6 Kmeans fitting with PCA score



A regression plot is used to investigate the likelihood of a relation after the fitting with PCA to determine if the selected 2 clusters were effective in taming kmeans for this mixture of data scales but still demonstrates that other models are more viable. The other visualization demonstrated the concatenation done with the PCA score results from the three components, onto the data frame before it fitted with a kmeans model. A segmentation to categorize the order of the scores in first, second third and fourth clusters as mapped onto the data frame. The resulting scatter plot shows how kmeans and PCA participate in separating the original two clusters and discovering more patterns in a categorical data frame

. Figure 7 4 new clusters from the Kmeans fitting with PCA scores

Figure 8 Regression plot for investigating if fitting the kmeans with PCA yields desired modelling

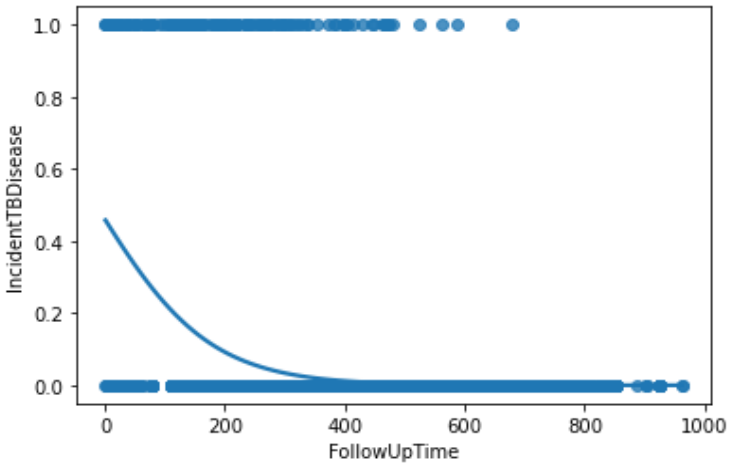


Figure 9 The 3 components added onto the dataset for fitting

```
In [357]: df_kmeans_pca
```

Out[357]:

Work	ContactHIV	IndexStrainGenotype	ContactDiabetes	Contact-IndexRoomshare	ContactPreviousTBHistory	MDR_or_SENS	Component 1	Component 2	Component3
2.0	0.0	4	0.0	0.0	0.0	2.0	1.314942	0.299571	-0.133735
3.0	0.0	4	0.0	0.0	0.0	2.0	1.227886	0.276486	0.008272
3.0	0.0	4	0.0	0.0	0.0	2.0	1.258137	0.505224	-0.453299
3.0	0.0	4	0.0	0.0	0.0	2.0	1.271805	0.288071	-0.068737
1.0	0.0	4	0.0	0.0	0.0	2.0	1.360635	0.554296	-0.702034
...	...	...	...	...	...	...	...	...	...
1.0	0.0	4	0.0	0.0	1.0	2.0	1.252744	-0.476657	-0.258430
2.0	0.0	8	0.0	0.0	0.0	2.0	1.276493	-0.627047	0.483973
2.0	0.0	8	0.0	0.0	0.0	2.0	1.279289	-0.620165	0.444958
3.0	0.0	8	0.0	0.0	0.0	2.0	1.220793	-0.652466	0.613269
3.0	0.0	8	0.0	1.0	0.0	2.0	1.242145	-0.467903	0.257881

However, as mentioned above and upon testing its accuracy only scored about 50%.

```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))

Result: 2079 out of 4117 samples were correctly labeled.
Accuracy score: 0.50
```

This means that the unsupervised learning techniques can allow for the investigation of any dataset, however analysts should keep in mind that their outcome is a binary scale. By using kmeans for investigations, an attainment of such accuracy results call for the application of superior techniques – supervised learning.

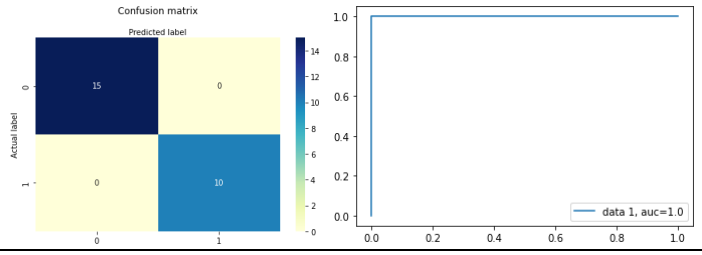
The Supervised learning techniques.

These are chosen based on the earlier findings that for binary outcome variables, it's safe to choose a statistical method that handles these variables and these are:

1. Logistic regression
2. Support vector machine
3. Random forests.

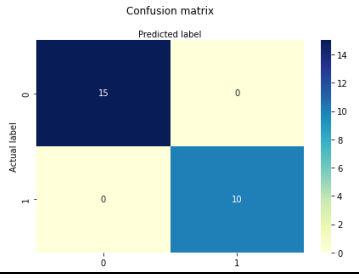
The major issues to report follow the methodology above in the sections of methods. Key outputs found below through visualizations and the rest added in the appendix section.

The Confusion matrix for the logistic regression with key outputs of the performance of the model showing 100% accuracy.



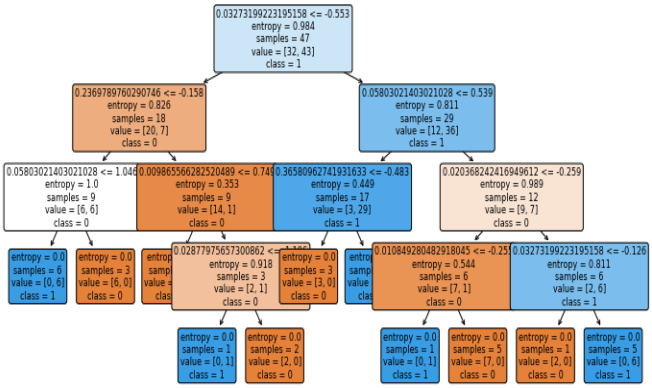
	precision	recall	f1-score	support
0	1.00	1.00	1.00	15
1	1.00	1.00	1.00	10
accuracy			1.00	25
macro avg	1.00	1.00	1.00	25
weighted avg	1.00	1.00	1.00	25

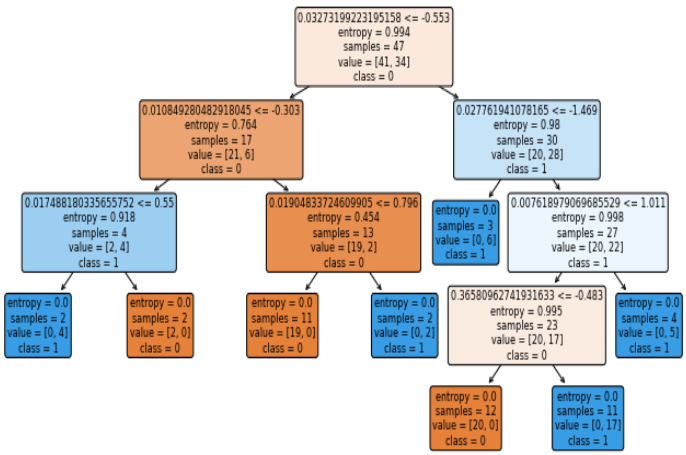
The Confusion matrix for the SVM with key outputs of the performance of the model showing 96% accuracy.

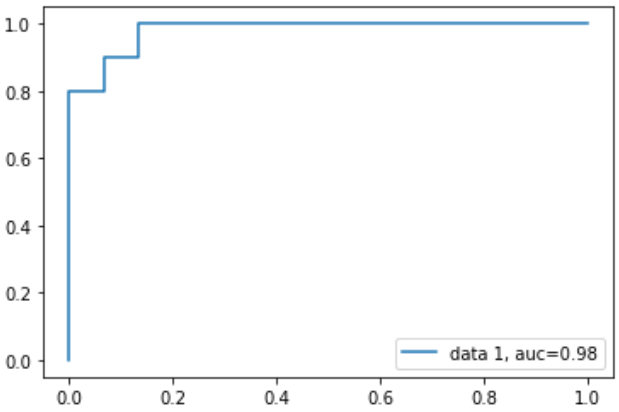


	precision	recall	f1-score	support
0	1.00	0.93	0.97	15
1	0.91	1.00	0.95	10
accuracy			0.96	25
macro avg	0.95	0.97	0.96	25
weighted avg	0.96	0.96	0.96	25

The Confusion matrix for the Random Forrest Classification with key outputs of the performance of the model showing 96% accuracy.







	precision	recall	f1-score	support
0	1.00	0.87	0.93	15
1	0.83	1.00	0.91	10
accuracy			0.92	25
macro avg	0.92	0.93	0.92	25
weighted avg	0.93	0.92	0.92	25

Discussion.

For categorical datasets as used in the assignment, several outcomes can be investigated. For the chosen techniques, logistic regression performs better than SVM and Random forest at 100%, meaning that it can be able to predict the outcome with another dataset of the same setting. Parameter tuning was not applied however scaling and normalizing the data yielded the best performance out of chosen techniques. A limitation to this work has been the time taken to wrangle the data and exploring other experimental.

Conclusion.

This assignment achieves it intention in providing techniques to predict a binary outcome such the presence of a disease and in this Tuberculosis in contacts of households. The products such as the code can be refined to fit other cases in public health such as chronic diseases that develop overtime.

References.

Grandjean, L., Gilman, R. H., Martin, L., Soto, E., Castro, B., Lopez, S., . . . Moore, D. A. J. (2015). Transmission of Multidrug-Resistant and Drug-Susceptible Tuberculosis within Households: A Prospective Cohort Study. *PLOS Medicine, 12*(6), e1001843. doi:10.1371/journal.pmed.1001843

Appendix.

Codes. For wrangling, supervised and unsupervised techniques

The Codes that were used to achieve this work.

SUPERVISED LEARNING

```
# In[1]:


from IPython import display

from sklearn.tree import export_graphviz

from sklearn import tree

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.model_selection import cross_val_score

from sklearn import model_selection

from sklearn.ensemble import RandomForestClassifier

from sklearn import svm

from sklearn import metrics

from sklearn.metrics import classification_report

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make_pipeline

from sklearn.datasets import make_classification

from sklearn.linear_model import LogisticRegression

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import mutual_info_classif

from sklearn.model_selection import train_test_split

from sklearn.feature_selection import chi2

import seaborn as sns

import sklearn

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.cluster import hierarchy

get_ipython().run_line_magic('matplotlib', 'inline')

# In[2]:

df = pd.read_csv("C:\\Users\\Personal\\Downloads\\Household_Study_Database.csv")

# In[3]:

df.head()

# In[4]
```

```
df.isnull()

# In[6]:

plt.figure(figsize=(10, 8))

sns.heatmap(df.isnull())

# In[7]:

print(df.dtypes)

# In[8]:

sns.countplot(x='Incident TB Disease', data=df)

# In[9]:

df['Incident TB Disease'].value_counts()

# In[10]:

df.isnull().sum()

# In[11]:

sns.countplot(x='Incident TB Disease', hue='Index Sex', data=df)

# In[12]:

sns.countplot(x='Incident TB Disease', hue='Contact Sex', data=df)

# In[13]:

df['Index Sputum Smear Grade'].value_counts()

# In[14]:

df.groupby('Incident TB Disease').count()
# In[15]:

df.describe(include='all')
# ### Imputation for missing values Since the data set has multiple data types of objects integres and float
we should corece to a format and replace all NaN values with appropriate methodds

# In[16]:

df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))

# In[17]:

df.head(20)

# In[18]:

df.isnull().sum()

# In[19]:

sns.countplot(x='Incident TB Disease', hue='Index Sex', data=df)

# In[20]:

sns.countplot(x='Index Education', data=df)

# In[21]:
```

```
df['Index Education'].value_counts()

# In[22]:

# In[23]:

y = df['MDR Sensitive Numeric']
X = df.drop(['MDR Sensitive Numeric'], axis=1)

# In[24]:

X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=100, test_size=0.3)

# In[25]:

cor = X_train.corr()
plt.figure(figsize=(18, 12))

sns.heatmap(cor, cmap=plt.cm.CMRmap_r, annot=True)

plt.show()



# ## But the correlation is useless for many features that are not linear continuos



# ### Instead use Chi2 test

# In[26]:

X_train = X_train.drop('MDR or Sensitive Household', axis=1)

# In[27]:

X_train

# In[28]:

f_p_values = chi2(X_train, y_train)

# In[29]:

f_p_values

# In[30]:

p_values = pd.Series(f_p_values[1])
p_values.index = X_train.columns
p_values

# In[31]:

p_values = p_values.sort_index(ascending=False)

# In[32]:

# determine the mutual information

mutual_info = mutual_info_classif(X_train, y_train)
mutual_info
```

```
# In[33]:

mutual_info = pd.Series(mutual_info)

mutual_info.index = X_train.columns

mutual_info.sort_values(ascending=False)


# In[34]:

mutual_info.sort_values(ascending=False).plot.bar(figsize=(20, 8))

# In[35]:

# No we Will select the top 5 important features

sel_five_cols = SelectKBest(mutual_info_classif, k=15)

sel_five_cols.fit(X_train, y_train)

X_train.columns[sel_five_cols.get_support()]

# In[36]:

X_train = X_train[['Family Code', 'Follow Up Time', 'Index Sputum Smear Grade',
                   'Index Hospitalization', 'Index Tobacco Use',
                   'Index Side Effects of Medication', 'Index History of TB Before',

                   'Index HIV Status', 'Index Cough Duration (days)', 'Index Work',

                   'Contact Chemotherapy', 'Contact Work', 'Index Strain Genotype',

                   'Contact-Index Roomshare', 'Contact Previous TB History']].copy()

# ## Logistic Regression Model

# In[37]:

X, y = make_classification(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
pipe = make_pipeline(StandardScaler(), LogisticRegression())
pipe.fit(X_train, y_train)

# In[38]:


logreg = LogisticRegression()

logreg.fit(X_train, y_train)

# In[39]:

prediction = logreg.predict(X_test)

# In[40]:

print(classification_report(y_test, prediction))

# In[41]:

cnf_matrix = metrics.confusion_matrix(y_test, prediction)
cnf_matrix

# In[42]:
```

```
class_names = [0, 1]  # name  of classes

fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))

plt.xticks(tick_marks, class_names)

plt.yticks(tick_marks, class_names)

# create heatmap

sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')

ax.xaxis.set_label_position("top")

plt.tight_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')


# In[43]:


y_pred_proba = logreg.predict_proba(X_test)[::, 1]

fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)

auc = metrics.roc_auc_score(y_test, y_pred_proba)

plt.plot(fpr, tpr, label="data 1, auc="+str(auc))

plt.legend(loc=4)

plt.show()

# ### Support Vector Machine on the Train and test data

# In[44]:

# Import svm model


# Create a svm Classifier
clf = svm.SVC(kernel='sigmoid')  # Linear Kernel

# Train the model using the training sets
clf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = clf.predict(X_test)

# In[45]:

# Import scikit-learn metrics module for accuracy calculation

# Model Accuracy: how often is the classifier correct?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# In[46]:

print(classification_report(y_test, y_pred))
```

```
# ### SVM performing at 96%,

# ## Random tree Classification  Model

# In[61]:

model = RandomForestClassifier(
    n_estimators=100, criterion='entropy', random_state=0)
model.fit(X_train, y_train)

# In[62]:

predictions = model.predict(X_test)
predictions

# In[63]:

model.classes_

# In[64]:

model.feature_importances_

# In[65]:

# In[66]:

# In[67]:


# ###### with cross validation of 10 folds

# In[68]:

model_cv_score = cross_val_score(model, X, y, cv=10, scoring='roc_auc')

# In[69]:

print(classification_report(y_test, predictions))

# ### Random Forest accuracy at 92%

# In[70]:

conf_matrix = metrics.confusion_matrix(y_test, prediction)
conf_matrix

# In[71]:

class_names = [0, 1]  # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(conf_matrix), annot=True, cmap="YlGnBu", fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

# In[72]:
```

```
y_pred_proba = model.predict_proba(X_test)[::, 1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr, tpr, label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()

# In[73]:

# Random Forest classification performed at 92%

# In[60]:

features = model.feature_importances_
classes = ['0', '1']
for estimator in model.estimators_:
    print(estimator)

    plt.figure(figsize=(12, 6))

    tree.plot_tree(estimator,

                   feature_names=features,

                   class_names=classes,

                   fontsize=8,

                   filled=True,

                   rounded=True)

plt.show()

fig.savefig('individualtree.png')



# In[ ]:

#!/usr/bin/env python
# coding: utf-8

# In[28]:


import sklearn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster import hierarchy


# In[3]:


import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


# In[5]:


df = pd.read_csv("C:\\Users\\Personal\\Downloads\\clean_HouseHoldTB.csv" )
```

```
# In[6]:


df


# In[9]:


print(df.dtypes)


# In[13]:


newData = df.copy()


# In[24]:


normalized_df=(newData-newData.min())/(newData.max()-newData.min())


# In[222]:


normalized_df


# # Hierarchical Clustering

# In[34]:



Z = hierarchy.linkage(normalized_df, 'ward', metric='euclidean')
hierarchy.dendrogram(Z, truncate_mode = 'level',p=4,

                     show_leaf_counts=False,
                     leaf_rotation=90.,leaf_font_size=12.,
    show_contracted=True) # -> you will have 4 leaf at the bottom of the plot
plt.title('Dendrogram1') # title of the dendrogram
plt.xlabel('IncidentTBDisease') # label of the x-axis
plt.ylabel('FamilyCode')
plt.show()


# In[32]:


dendrogram = sch.dendrogram(sch.linkage(normalized_df, method = 'ward'))
plt.title('Dendrogram2') # title of the dendrogram
plt.xlabel('IncidentTBDisease') # label of the x-axis
plt.ylabel('FollowUpTime') # label of the y-axis
plt.axhline(y=15000, c='grey', lw=1, linestyle='dashed')
plt.show()


# In[135]:
```

```
Z = hierarchy.linkage(normalized_df, 'ward')
hierarchy.dendrogram(Z, truncate_mode = 'level',p=5,

                     show_leaf_counts=False,
                     leaf_rotation=90.,leaf_font_size=12.,
    show_contracted=True) # -> you will have 4 leaf at the bottom of the plot
plt.title('Dendrogram3') # title of the dendrogram
plt.xlabel('MDRSensitiveNumeric') # label of the x-axis
plt.ylabel('FamilyCode')
plt.show()


# In[44]:



Z = hierarchy.linkage(normalized_df, 'ward')
hierarchy.dendrogram(Z, truncate_mode = 'level',p=5,

                     show_leaf_counts=False,
                     leaf_rotation=90.,leaf_font_size=12.,
    show_contracted=True) # -> you will have 4 leaf at the bottom of the plot
plt.title('Dendrogram3') # title of the dendrogram
plt.xlabel('sample index') # label of the x-axis
plt.ylabel('distance')
plt.show()


# In[421]:


plt.figure(figsize=(10,5))
Z = hierarchy.linkage(normalized_df, 'ward', metric='euclidean')
hierarchy.dendrogram(Z, truncate_mode = 'level',p=5,

                     show_leaf_counts=False,
                     leaf_rotation=90.,leaf_font_size=12.,
    show_contracted=True) # -> you will have 4 leaf at the bottom of the plot
plt.title('Dendrogram1') # title of the dendrogram
plt.ylabel('IncidentTBDisease') # label of the x-axis
plt.xlabel('MDRSensitiveNumeric')
plt.show()


# In[65]:


from sklearn.cluster import AgglomerativeClustering

Agg_hc = AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', linkage = 'ward')
y_hc = Agg_hc.fit_predict(normalized_df)


# In[70]:


print(cluster_labels)


# In[339]:


plt.figure(figsize=(16,5))
Z = hierarchy.linkage(normalized_df, 'ward', metric='euclidean')
```

```
hierarchy.dendrogram(Z, truncate_mode = 'level',p=5,

                     show_leaf_counts=False,
                     leaf_rotation=90.,leaf_font_size=12.,
    show_contracted=True) # -> you will have 4 leaf at the bottom of the plot

plt.title('Dendrogram1') # title of the dendrogram

plt.ylabel('IndexStrainGenotype') # label of the x-axis
plt.xlabel('IncidentTBDisease')

plt.show()


# ## AgglomerativeCllustering with PCA n =3

# In[418]:


from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt


# Perform Agglomerative Clustering
agglomerative_clustering = AgglomerativeClustering(n_clusters=2)
cluster_labels = agglomerative_clustering.fit_predict(normalized_df)

# Apply PCA to reduce the dimensionality of the data to 2 dimensions
pca = PCA(n_components= 3)
pca_transformed_data = pca.fit_transform(normalized_df)


# Visualize the clusters in a scatter plot
plt.figure(figsize=(15,6))
plt.scatter(pca_transformed_data[:, 0], pca_transformed_data[:, 1], c=cluster_labels)

plt.show()


# In[323]:


pca_transformed_data


# In[324]:


scores_pca = pca_transformed_data


# # Use kmeans with PCA scores

# In[383]:


##option1
wcss=[]
for i in range(1,21):
        kmeans_pca = KMeans(n_clusters = i, init ='k-means++', random_state=42)
        kmeans_pca.fit(scores_pca)
        wcss.append(kmeans_pca.inertia_)
```

```
# In[384]:


kmeans_pca.labels_


# In[387]:


kmeans_pca = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
kmeans_pca.fit(scores_pca)


# In[327]:


plt.figure(figsize=(10,8))
plt.plot(range(1,21),wcss, marker = 'o', linestyle = '--')
plt.xlabel('Number of Clusters')
plt.ylabel('K-means with PCA Clustering')
plt.show()


# In[342]:


normalized_df


# In[349]:


#analysis of Kmeans with PCA

df_kmeans_pca = pd.concat([df.reset_index(drop=True), pd.DataFrame(scores_pca)], axis=1)


# In[354]:


df_kmeans_pca= df_kmeans_pca.drop(['Segment K_means PCA'], axis= 1)


# In[355]:


df_kmeans_pca


# In[356]:


df_kmeans_pca.columns.values[-3:]= ['Component 1','Component 2', 'Component3']


# In[357]:


df_kmeans_pca


# In[388]:
```

```
df_kmeans_pca['Segment K-means PCA'] = kmeans_pca.labels_


# In[389]:


df_kmeans_pca


# In[390]:


df_kmeans_pca['Segment']                =                df_kmeans_pca['Segment                K-means
PCA'].map({0:'first',1:'second',2:'third',3:'fourth'})


# In[398]:


df_kmeans_pca


# # Plotting Clusters with Kmeans and PCA score

# In[399]:


x_axis = df_kmeans_pca['Component 2']
y_axis = df_kmeans_pca['Component 1']
plt.figure(figsize = (10,8))
sns.scatterplot(x_axis,y_axis, hue = df_kmeans_pca['Segment'], palette = ['g','r','c','m'])
plt.title('Clusters by PCA and KMeans')
plt.show()


# In[ ]:

# ## KMeans without PCA

# In[315]:

##option2
from sklearn import preprocessing
kmData = df.copy()
scaler = preprocessing.MinMaxScaler()
scaler.fit_transform(normalized_df)

# In[329]:


from sklearn.cluster import KMeans
#checking for number of clusters to select using elbow method
wcss=[]
for i in range(1,7):
        kmeans = KMeans(i)
        kmeans.fit(kmData)
        wcss_iter = kmeans.inertia_
        wcss.append(wcss_iter)

number_clusters = range(1,7)
plt.figure(figsize=(10,8))
plt.plot(number_clusters,wcss)
plt.title('The Elbow title')
```

```
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')


# ## still shows only 2 clusters can be gotten

# ## choosing 2 clusters for the model with a kink

# In[319]:

#cheking correct predictions with SputumSmearGrade at n clusters
#x = kmData.drop(['IncidentTBDisease'], axis =1)
#y = kmData['FollowUpTime']
kmeans = KMeans(n_clusters=2, init = 'k-means++', random_state=42)

kmeans.fit(kmData)

labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))


# ### unsupervised learning only scoring 50% of correctly labeled features, underperforming kmeans

# ## plotting fatures to investigate relationships

# In[281]:

plt.figure(figsize = (12,9))
plt.scatter(df.iloc[:, 4], df.iloc[:,5])
plt.xlabel('FollowupTime')
plt.ylabel('Sex')

plt.title('Checking distribution of sex with follow up time')

# In[345]:

import seaborn as sns
x = df['FollowUpTime']
y = df['IncidentTBDisease']
sns.regplot(x=x, y=y, data=normalized_df, logistic=True, ci=None)
# In[414]:
sns.lmplot(x='FollowUpTime', y= 'IncidentTBDisease', data = newData, fit_reg=False)


# In[251]:


sns.scatterplot( x='FollowUpTime', y= 'CoughDuration(days)', data=normalized_df, hue ='ContactSex')


# In[252]:


sns.scatterplot( x='FollowUpTime', y= 'CoughDuration(days)', data=kmData, hue ='ContactSex')


# In[411]:
```

```
sns.scatterplot( x='FollowUpTime', y= 'IncidentTBDisease', data=kmData, hue ='ContactHIV')


# # logistic regression supervised

# In[256]:
y= df['IncidentTBDisease']
x = df.drop(['IncidentTBDisease'], axis =1)


# In[258]:
from sklearn.linear_model import LogisticRegression


# In[259]:
from sklearn.model_selection import train_test_split


# In[260]:
y= df['IncidentTBDisease']
X = df.drop(['IncidentTBDisease'], axis =1)
X_train, X_test,y_train, y_test = train_test_split(X,y,test_size=0.33, random_state=42)


# In[266]:
from sklearn.datasets import make_classification
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
X, y = make_classification(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
pipe = make_pipeline(StandardScaler(), LogisticRegression())
pipe.fit(X_train, y_train)


# In[267]:


logreg = LogisticRegression()
logreg.fit(X_train,y_train)


# In[271]:


prediction =logreg.predict(X_test)


# In[272]:


from sklearn.metrics import classification_report

print(classification_report(y_test,prediction))


# ## Prediction of 100%, very questionable but fits since the data is categorical. Overfitting detected due
to lack of feature selection

# #### test with confusion matrix

# In[424]:
```

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, prediction)
cnf_matrix


# ### Plotting the confusion matrix

# In[412]:


import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')


# ### ROC of the overfitted data

# In[409]:


y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()


# In[400]:


df                                          =                                           pd.get_dummies(df,
columns=["IncidentTBDisease","FollowUpTime","Sex","Education","SputumSmearGrade","Diabetes",
 "Incarceration","Hospitalization","AlcoholUse","TobaccoUse","SideEffectsofMedication",
        "HistoryofTBBefore",
            "HIVStatus","CoughDuration(days)","Work","MDRSensitiveNumeric","SocioEconomicTertile",
])

# # Supervised learning starts here

# In[425]:

pip install dtreeviz

# In[ ]:
```