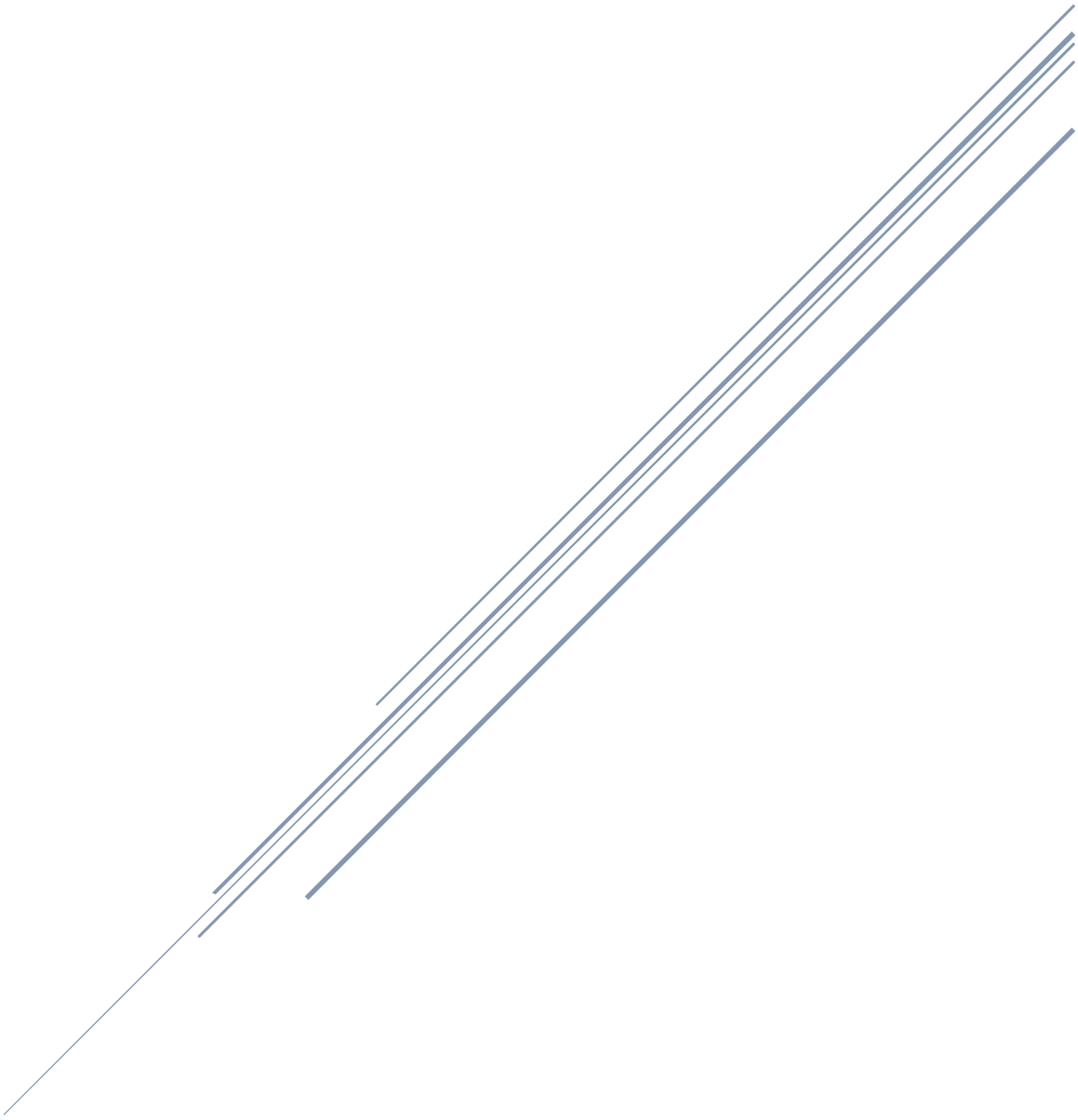
Smart Health Disease Prediction.

DataMining Project.



Name : Farah Essam Eldeen Sayed Elsayed Ahmed.

ID: 20221462956

Name : Menna Allah Mohamed Elsayed.

ID: 20221462499

Name : Mohamed Ragb Mohamed Mousa.

ID: 20221454736

Name : Tarek Mahmoud Zaki.

ID: 20221374036

Name : Elmoatasem Billah Mohamed Mofadal .

ID: 20221372508

Project Idea:

Nowadays, medical care is something that anyone might need immediately, but unavailable due to various reasons. Smart health disease prediction is an end user support system that allows users to get guidance immediately with the help of an online intelligent health system. The system holds complete information about symptoms and the diseases associated with it. The system analyzes diseases associated with the symptoms for the patient and advises them for X-ray, blood test or CT scan as requested by the system. Users can also directly get in touch with the specialist doctors for any ailment and share their reports. It is not just one time, rather a proper login detail is shared for future use.

First we import the library that we will use in the project:

```
In [1]: import pandas as pd
        import numpy as np
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split,cross_val_score
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.metrics import accuracy_score
        from sklearn.cluster import KMeans
        import matplotlib.pyplot as plt
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix,ConfusionMatrixDisplay, classification_report
        import seaborn as sns
```

We read the TrainingData and the TestingData using the pandas library and droping a useless column named 'Unnamed column' from the TrainingData .

```
In [6]: TrainingData = TrainingData.drop(["Unnamed: 133"],axis=1)
```

Now we will check if is there values smaller than zero

```
In [7]: train=TrainingData.drop(['prognosis'],axis=1)
        (train[train.columns]<0).sum()

Out[7]: itching                 0
        skin_rash               0
        nodal_skin_eruptions    0
        continuous_sneezing     0
        shivering               0
                               ..
        small_dents_in_nails    0
        inflammatory_nails      0
        blister                 0
        red_sore_around_nose    0
        yellow_crust_ooze       0
        Length: 132, dtype: int64

In [8]: test=TestingData.drop(['prognosis'],axis=1)
        (test[test.columns]<0).sum()

Out[8]: itching                 0
        skin_rash               0
        nodal_skin_eruptions    0
        continuous_sneezing     0
        shivering               0
                               ..
        small_dents_in_nails    0
        inflammatory_nails      0
        blister                 0
        red_sore_around_nose    0
        yellow_crust_ooze       0
        Length: 132, dtype: int64
```

we will check if is there null values and the shape of the data after cleaning

```
In [10]: TrainingData.isnull().sum()

Out[10]: itching                 0
         skin_rash               0
         nodal_skin_eruptions    0
         continuous_sneezing     0
         shivering               0
                                ..
         inflammatory_nails      0
         blister                 0
         red_sore_around_nose    0
         yellow_crust_ooze       0
         prognosis               0
         Length: 133, dtype: int64

In [11]: TestingData.isnull().sum()

Out[11]: itching                 0
         skin_rash               0
         nodal_skin_eruptions    0
         continuous_sneezing     0
         shivering               0
                                ..
         inflammatory_nails      0
         blister                 0
         red_sore_around_nose    0
         yellow_crust_ooze       0
         prognosis               0
         Length: 133, dtype: int64

In [12]: TestingData.shape

Out[12]: (42, 133)

In [13]: TrainingData.shape

Out[13]: (4920, 133)
```

This dataset is a clean dataset with no null values and all the features consist of 0's and 1s and the dataset is a balanced dataset, There are exactly 120 samples for each disease.

do a function called get_sub_dataframe takes two argument ,first the function initialize two empty list val and fet to store the frequency count and column name to select row from data we used variable called df_after_selection , we created the val_df to access columns in data, select all columns and assingns the resulting and we don't select tha last column cause we want to count the frequency of each non-zero value . mat created by iterating over the columns of val_df and creating list of tuples contains frequency count and column name for each non-zero value in column,tuples created for the column sum of values is non-zero, val and fat list populted with frequency count and column name from each tuple in the mat list respectively,the data variable created as a dictionary maps the fet list to the 'fet' key and the val list to the 'val' key,data dictionary is coverted to pandas,and the resulting dataframe is returned as the output of the function.

In [42]:
```
def get_sub_dataframe(TrainingData:pd.DataFrame, prognosis:str) -> pd.DataFrame:
    val, fet = [],[]
    df_after_selection = TrainingData[TrainingData['prognosis']== prognosis ]
    val_df = df_after_selection.iloc[:,:-1]
    mat = [[val_df[i].sum(),i] for i in val_df.columns if val_df[i].sum()!=0]
    for i in mat:
        val.append(i[0])
        fet.append(i[1])
    data = {'fet': fet, 'val': val}
    data = pd.DataFrame.from_dict(data)
    return data
```

It's time to split the data to train and test the model. We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models.

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[16]:
```
((3936, 132), (984, 132), (3936,), (984,))
```

After splitting the data, we will be now working on the modeling part. We will be using K-Fold cross-validation to evaluate the machine-learning models. We will be using Gaussian Naive Bayes Classifier, and Random Forest Classifier for cross-validation . cross-validation is a robust measure to prevent overfitting.

K-Fold cross-validation is one of the cross-validation techniques in which the whole dataset is split into k number of subsets, also known as folds, then training of the model is performed on the k-1 subsets , and the remaining one subset is used to evaluate the model performance.we need to partition the data into k folds. Then ,we iteratively train the algorithm on k-1 folds while using the remaining holdout fold as the test set, the function cv_score is evaluate the score of the model take the (estimator (model), X, y) .

In [18]:
```
def cv_scoring(estimator, X, y):
    return accuracy_score(y, estimator.predict(X))

# Initializing Models
models = {
    "Gaussian NB":MultinomialNB(),
    "Random Forest":RandomForestClassifier(random_state=18)
}

# Producing cross validation score for the models
for model_name in models:
    model = models[model_name]
    scores = cross_val_score(model, X, y, cv = 10,     #10 k_fold
                             scoring = cv_scoring)
    print("=="*30)
    print(model_name)
    print(f"Scores: {scores}")
    print(f"Mean Score: {np.mean(scores)}")
#From the above output, we can notice that all our machine learning algorithms are performing very well
#This approach will help us to keep the predictions much more accurate on completely unseen data.
```

the output Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.] Mean Score: 1.0 , we can notice that all our machine learning algorithms are performing very well ,This approach will help us to keep the predictions much more accurate on completely unseen data.

Now we will be training the two models on the train data:

we train the Naive Bayes classifier using MultinomialNB from scikit-learn and fit it to the training data.

```
clf = MultinomialNB()
clf.fit(X_train, y_train)
```

Out[20]:
```
MultinomialNB()
```

In [21]:
```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))
```

```
Accuracy: 100.00%
```

In [22]:
```
new_data = pd.DataFrame(X_test)
prediction = clf.predict(new_data)
print('Predicted disease:', prediction[3])
```

```
Predicted disease: AIDS
```

Confusion matrices are generally used only with class output models, A confusion matrix is an N X N matrix, where N is the number of predicted classes. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix. It is a performance measurement for machine learning classification problems where the output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. It is extremely useful for measuring precision-recall, Specificity, Accuracy, and most importantly, AUC-ROC curves.

checking the quality of our model NB using a confusion matrix .

In [23]:
```
cf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(12,8))
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Naive Bayes Classifier on Test Data")
plt.show()
```

classification_report NB :

```
In [24]: print(classification_report(y_test, y_pred))
         print(accuracy_score(y_test, y_pred))
```

```
                                        precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00        18
                                AIDS          1.00      1.00      1.00        30
                                Acne          1.00      1.00      1.00        24
                Alcoholic hepatitis           1.00      1.00      1.00        25
                             Allergy          1.00      1.00      1.00        24
                           Arthritis          1.00      1.00      1.00        23
                   Bronchial Asthma           1.00      1.00      1.00        33
               Cervical spondylosis           1.00      1.00      1.00        23
                        Chicken pox           1.00      1.00      1.00        21
               Chronic cholestasis            1.00      1.00      1.00        15
                        Common Cold           1.00      1.00      1.00        23
                             Dengue           1.00      1.00      1.00        26
                           Diabetes           1.00      1.00      1.00        21
          Dimorphic hemmorhoids(piles)        1.00      1.00      1.00        29
                        Drug Reaction         1.00      1.00      1.00        24
                    Fungal infection          1.00      1.00      1.00        19
                                GERD          1.00      1.00      1.00        28
                     Gastroenteritis          1.00      1.00      1.00        25
                        Heart attack          1.00      1.00      1.00        23
                          Hepatitis B         1.00      1.00      1.00        27
                          Hepatitis C         1.00      1.00      1.00        26
                          Hepatitis D         1.00      1.00      1.00        23
                          Hepatitis E         1.00      1.00      1.00        29
                         Hypertension         1.00      1.00      1.00        25
                        Hyperthyroidism       1.00      1.00      1.00        24
                          Hypoglycemia        1.00      1.00      1.00        26
                        Hypothyroidism        1.00      1.00      1.00        21
                             Impetigo         1.00      1.00      1.00        24
                             Jaundice         1.00      1.00      1.00        19
                              Malaria         1.00      1.00      1.00        22
                             Migraine         1.00      1.00      1.00        25
                       Osteoarthristis        1.00      1.00      1.00        22
          Paralysis (brain hemorrhage)        1.00      1.00      1.00        24
                  Peptic ulcer diseae         1.00      1.00      1.00        17
                             Pneumonia        1.00      1.00      1.00        28
                             Psoriasis        1.00      1.00      1.00        22
                           Tuberculosis       1.00      1.00      1.00        25
                              Typhoid         1.00      1.00      1.00        19
       Urinary tract infection               1.00      1.00      1.00        26
                        Varicose veins        1.00      1.00      1.00        22
                           hepatitis A        1.00      1.00      1.00        34

                              accuracy                           1.00       984
                             macro avg       1.00      1.00      1.00       984
                          weighted avg       1.00      1.00      1.00       984

1.0
```

Fitting the model NB the Test dataset:

```
In [26]: Test_data = TestingData.drop(["prognosis"],axis=1)
         y_testData=TestingData["prognosis"]
         test1=TestingData.join(pd.DataFrame(clf.predict(Test_data),columns=["predicted"]))[["prognosis","predicted"]]
         test1
```

The function accuracy score tell the accuracy of the model , here the accuracy of the model NB in the test data is 100%

```
In [27]: accuracyTest = accuracy_score(y_testData, clf.predict(Test_data))
         print("accuracyTest: {:.2f}%".format(accuracyTest*100))
```

```
accuracyTest: 100.00%
```

we train the RandomForestClassifier and fit it to the training data.

```
In [28]: classifier = RandomForestClassifier()
         classifier.fit(X_train, y_train)

Out[28]: RandomForestClassifier()
```

```
In [29]: y_pred2 = classifier.predict(X_test)
         accuracy2 = accuracy_score(y_test, y_pred2)
         print("Accuracy: {:.2f}%".format(accuracy2*100))

         Accuracy: 100.00%
```

```
In [30]: prediction2 = classifier.predict(new_data)
         print('Predicted disease:', prediction2[3])

         Predicted disease: AIDS
```

checking the quality of our model RandomForestClassifier using a confusion matrix .

```
In [31]: cf_matrix2 = confusion_matrix(y_test, y_pred2)
         plt.figure(figsize=(12,8))
         sns.heatmap(cf_matrix2, annot=True)
         plt.title("Confusion Matrix for RandomForestClassifie on Test Data2")
         plt.show()
```

classification_report RandomForestClassifier

```
In [32]: print(classification_report(y_test, y_pred2))
         print(accuracy_score(y_test, y_pred2))
```

	precision	recall	f1-score	support
(vertigo) Paroymsal Positional Vertigo	1.00	1.00	1.00	18
AIDS	1.00	1.00	1.00	30
Acne	1.00	1.00	1.00	24
Alcoholic hepatitis	1.00	1.00	1.00	25
Allergy	1.00	1.00	1.00	24
Arthritis	1.00	1.00	1.00	23
Bronchial Asthma	1.00	1.00	1.00	33
Cervical spondylosis	1.00	1.00	1.00	23
Chicken pox	1.00	1.00	1.00	21
Chronic cholestasis	1.00	1.00	1.00	15
Common Cold	1.00	1.00	1.00	23
Dengue	1.00	1.00	1.00	26
Diabetes	1.00	1.00	1.00	21
Dimorphic hemmorhoids(piles)	1.00	1.00	1.00	29
Drug Reaction	1.00	1.00	1.00	24
Fungal infection	1.00	1.00	1.00	19
GERD	1.00	1.00	1.00	28
Gastroenteritis	1.00	1.00	1.00	25
Heart attack	1.00	1.00	1.00	23
Hepatitis B	1.00	1.00	1.00	27
Hepatitis C	1.00	1.00	1.00	26
Hepatitis D	1.00	1.00	1.00	23
Hepatitis E	1.00	1.00	1.00	29
Hypertension	1.00	1.00	1.00	25
Hyperthyroidism	1.00	1.00	1.00	24
Hypoglycemia	1.00	1.00	1.00	26
Hypothyroidism	1.00	1.00	1.00	21
Impetigo	1.00	1.00	1.00	24
Jaundice	1.00	1.00	1.00	19
Malaria	1.00	1.00	1.00	22
Migraine	1.00	1.00	1.00	25
Osteoarthristis	1.00	1.00	1.00	22
Paralysis (brain hemorrhage)	1.00	1.00	1.00	24
Peptic ulcer diseae	1.00	1.00	1.00	17
Pneumonia	1.00	1.00	1.00	28
Psoriasis	1.00	1.00	1.00	22
Tuberculosis	1.00	1.00	1.00	25
Typhoid	1.00	1.00	1.00	19
Urinary tract infection	1.00	1.00	1.00	26
Varicose veins	1.00	1.00	1.00	22
hepatitis A	1.00	1.00	1.00	34
accuracy			1.00	984
macro avg	1.00	1.00	1.00	984
weighted avg	1.00	1.00	1.00	984

```
1.0
```

Fitting the model RandomForestClassifier the Test dataset:

```
In [34]: test2=TestingData.join(pd.DataFrame(classifier.predict(Test_data),columns=["predicted"]))[["prognosis","predicted"]]
         test2
```

the accuracy of the model RandomForestClassifier in the test data is 97.62%

```
In [35]: accuracyTest2 = accuracy_score(y_testData, classifier.predict(Test_data))
         print("accuracyTest2: {:.2f}%".format(accuracyTest2*100))

         accuracyTest2: 97.62%
```

If we use SVC (Support Vector Classifier) to fit the test data set we get accuracy 100%

```
In [38]: accuracyTest3 = accuracy_score(y_testData, svm_model.predict(Test_data))
         print("accuracyTest3: {:.2f}%".format(accuracyTest3*100))

         accuracyTest3: 100.00%
```

even one of the models makes wrong predictions ,and the other two make correct predictions then the final output would be the correct one

we conclude that RandomForestClassifier is doesnt predict good enough for our data