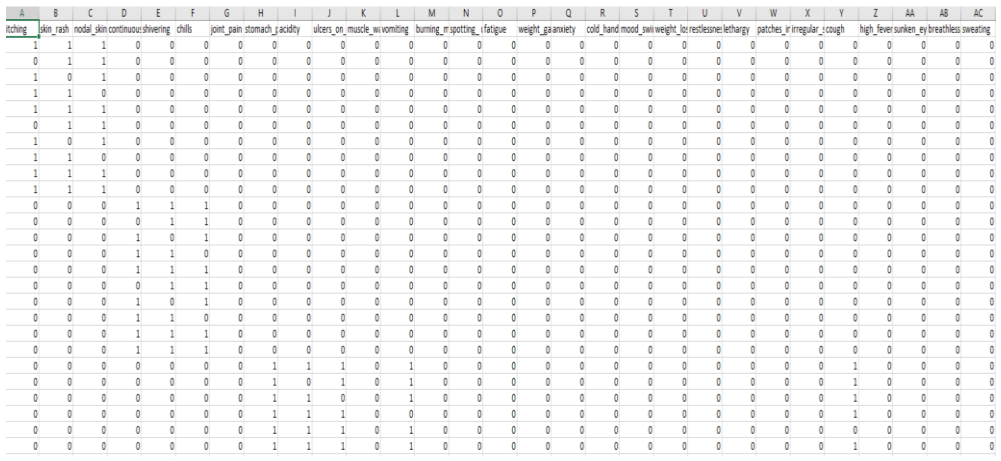
DISEASE PREDICTION MODEL DEPLOYMENT WITH IBM CLOUD WATSON STUDIO

INTRODUCTION:

In our fast-paced world, technology is working behind the scenes to help us in ways we might not even realize. Imagine if we could predict diseases earlier and more accurately, allowing doctors to provide better care. That's where our story begins. We've used machine learning algorithms to create a special tool that helps predict diseases. It's like having a super-smart assistant for doctors, and we have deployed it in IBM Cloud Watson Studio. Let’s now see the implementation details of this project.

DATASET USED:

I have used the following dataset to train my model. The link for the dataset is [Disease Prediction Using Machine Learning (kaggle.com)](kaggle.com).



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
itching	skin_rash	nodal_skin	continuous	shivering	chills	joint_pain	stomach_	acidity	ulcers_on_	muscle_wa	vomiting	burning_m	spotting_	fatigue	weight_ga	anxiety	cold_hand	mood_swi	weight_los	restlessnes	lethargy	patches_ir	irregular_s	cough	high_fever	sunken_ey	breathless	sweating
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

PREDICTIVE USE CASE:

Here are some common predictive use cases:

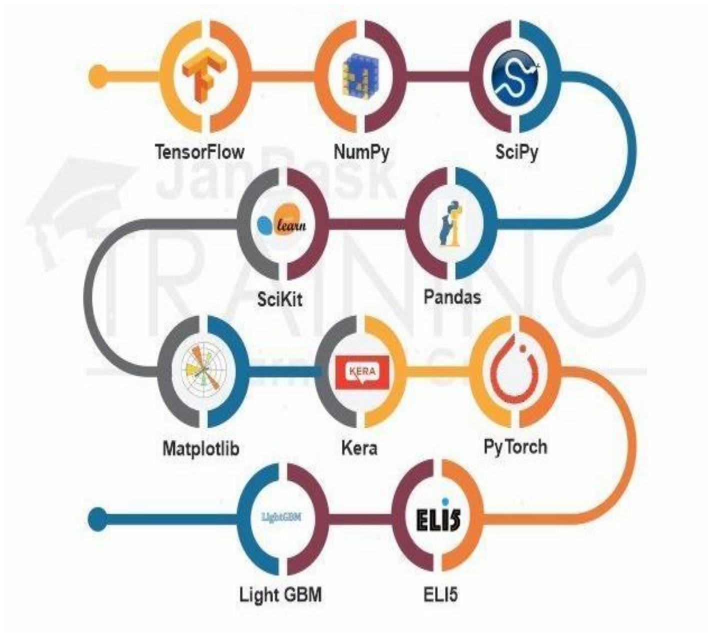
Disease Risk Assessment: You can use your model to predict an individual's risk of developing a specific disease, such as heart disease, diabetes, or cancer. By analyzing a patient's medical history, lifestyle, and genetic factors, the model can provide risk scores and recommendations for preventive measures.

Early Disease Detection: Deploying a disease prediction model can help with early detection of diseases like cancer or infectious diseases. The model can analyze medical imaging data, lab results, or patient symptoms to identify potential health issues at an early stage, improving treatment outcomes.

Chronic Disease Management: Patients with chronic conditions, such as asthma or diabetes, can benefit from predictive models that monitor their health and provide personalized recommendations for managing their conditions. The model can analyze patient data and alert healthcare providers or patients when intervention is needed.

Medication Adherence: Predictive models can assist in medication adherence. By analyzing a patient's history and behavior, the model can predict the likelihood of a patient adhering to a prescribed medication regimen and provide reminders or interventions when adherence is at risk.

IMPORTING NECESSARY LIBRARIES:



```
# Importing libraries
import numpy as np
import pandas as pd
import warnings
!pip install scipy==1.9.0
from scipy.stats import mode
from sklearn.preprocessing import
LabelEncoder
from sklearn.model_selection import
train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.naive_bayes import
GaussianNB
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

numpy (as np): NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays.

pandas (as pd):Pandas is a powerful library for data manipulation and analysis. It provides data structures such as DataFrames and Series, which allow you to easily read, manipulate, and analyze tabular data.

warnings: The warnings module is part of Python's standard library and allows you to control how warnings are displayed and handled in your code. You can use it to filter or suppress specific warnings.

scipy.stats.mode: The mode function is part of the SciPy library, which builds on NumPy and provides additional scientific and technical computing functionality. The mode function calculates the mode of a dataset, which is the most frequently occurring value.

sklearn.preprocessing.LabelEncoder: LabelEncoder is a class from the scikit-learn library (sklearn) used for encoding categorical variables into numerical values. It assigns a unique integer to each category in a categorical feature.

sklearn.model_selection: The model_selection module within scikit-learn provides functions and classes for model selection, including data splitting, cross-validation, and hyperparameter tuning.

sklearn.svm.SVC: SVC stands for Support Vector Classifier, and it's a class within scikit-learn for building Support Vector Machine (SVM) classifiers. SVM is a powerful machine learning algorithm for classification and regression tasks.

sklearn.naive_bayes.GaussianNB: GaussianNB is a class within scikit-learn for implementing the Gaussian Naive Bayes classifier. Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem.

sklearn.ensemble.RandomForestClassifier: RandomForestClassifier is a class within scikit-learn for building random forest models. Random forests are an ensemble learning technique that combines multiple decision trees to improve predictive accuracy.

sklearn.metrics: The metrics module within scikit-learn provides various metrics and evaluation functions to assess the performance of machine learning models.

```
# Reading the train.csv by removing the
# last column since it's an empty column

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Vc2zdhq_roalu42Y6nSRweuUu6v5LKoltXck2ft2GP2F',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu-de.cloud-object-storage.appdomain.cloud')

bucket = 'diseasepredictionmodeldeployment-donotdelete-pr-1rvaykcaofzaak'
object_key = 'Training.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

data= pd.read_csv(body).dropna(axis = 1)
data.head()


disease_counts = data["prognosis"].value_counts()
temp_df = pd.DataFrame({
"Disease": disease_counts.index,
"Counts": disease_counts.values
})
```

In [4]:

```
# Encoding the target value into numerical
# value using LabelEncoder
encoder = LabelEncoder()
data["prognosis"] = encoder.fit_transform(data["prognosis"])
```

In [5]:

```
X = data.iloc[:,:-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test =train_test_split(
X, y, test_size = 0.2, random_state = 24)

print(f"Train: {X_train.shape}, {y_train.shape}")
print(f"Test: {X_test.shape}, {y_test.shape}")
Train: (3936, 132), (3936,)
Test: (984, 132), (984,)
```

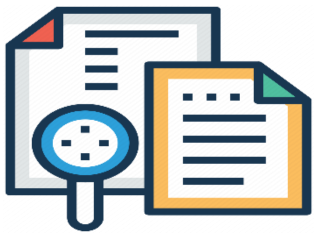
In [6]:

```
# Defining scoring metric for k-fold cross validation
def cv_scoring(estimator, X, y):
        return accuracy_score(y, estimator.predict(X))

# Initializing Models
models = {
        "SVC":SVC(),
        "Gaussian NB":GaussianNB(),
        "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,
                                                        n_jobs = -1,
                                                        scoring = cv_scoring)
        print("=="*30)
        print(model_name)
        print(f"Scores: {scores}")
        print(f"Mean Score: {np.mean(scores)}")
============================================================
SVC
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Gaussian NB
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
============================================================
Random Forest
Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean Score: 1.0
```

The detailed explanation of the above code is:



Data Retrieval:

The code retrieves a CSV file named Training.csv from IBM Cloud Object Storage using the IBM Cloud Python SDK (ibm_boto3).

Data Preprocessing:

It loads the CSV data into a Pandas DataFrame.

Drops any columns that contain missing (NaN) values to ensure data cleanliness.

Creates a DataFrame called temp_df to count the occurrences of different diseases in the "prognosis" column.

Encodes the target variable "prognosis" into numerical values using the LabelEncoder from scikit-learn.

Data Splitting:

Splits the data into training and testing sets using the train_test_split function from scikit-learn.

80% of the data is used for training (X_train and y_train), and 20% is used for testing (X_test and y_test).

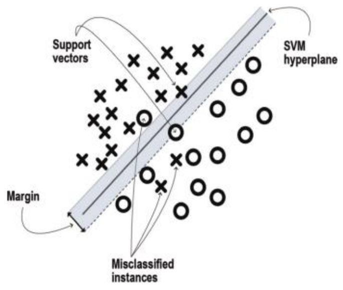
Cross-Validation and Model Evaluation:

Defines a custom scoring metric function, cv_scoring, which calculates accuracy using accuracy_score from scikit-learn. This metric will be used for cross-validation.

Model Initialization and Cross-Validation:

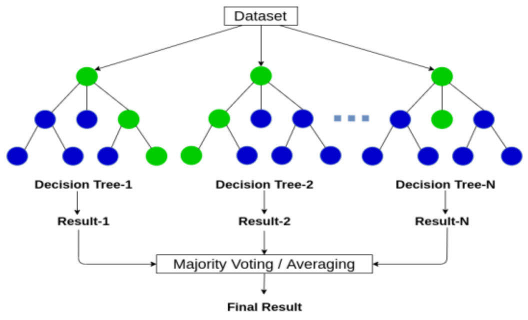
Initializes three machine learning models:

Support Vector Classifier (SVC): A classification algorithm that attempts to find an optimal hyperplane to separate different classes in the data.



Gaussian Naive Bayes (GaussianNB): A probabilistic classification algorithm based on Bayes' theorem, assuming Gaussian (normal) distribution of data.

Random Forest Classifier (RandomForestClassifier): An ensemble learning method that combines multiple decision trees to improve predictive accuracy.



Performs 10-fold cross-validation for each model using the cross_val_score function. For each fold, the accuracy scores are computed and printed.

Additionally, the mean accuracy score across all folds is calculated and displayed for each model.

In summary, this code loads a dataset, preprocesses it by handling missing values and encoding the target variable, splits the data into training and testing sets, and evaluates three machine learning models using cross-validation. The models are evaluated in terms of their accuracy, and the results are displayed, indicating how well each model performs on the given dataset. The code is a starting point for building and evaluating machine learning models for disease prediction.