Studying about different types of algorithms used in AI and machine learning. Comparing the accuracy rate among the below given certain algorithms used in machine learning.

Here is an overview of the various algorithms:-

1)Random Forest Algorithm:

Random Forest is an ensemble learning method primarily used for classification and regression tasks. It works by constructing multiple decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees.

How it Works:

1. **Bootstrapping**: Multiple subsets of the training data are created by random sampling with replacement.
2. **Decision Trees**: For each subset, a decision tree is constructed. While splitting nodes in these trees, a random subset of features is considered, ensuring diversity among the trees.
3. **Aggregation**: The final prediction is made by aggregating the predictions of all individual trees. For classification, it is usually done by majority vote; for regression, by averaging the predictions.

Advantages:

- **Robustness**: It reduces overfitting by averaging multiple trees, thus improving generalization.
- **Feature Importance**: It provides a way to assess feature importance.
- **Versatility**: Applicable to both classification and regression tasks.

Disadvantages:

- **Complexity**: The model can be complex and computationally intensive.
- **Interpretability**: Individual trees are easy to interpret, but the ensemble (forest) is less interpretable.

2)XGBoost Algorithm:

XGBoost, which stands for eXtreme Gradient Boosting, is an advanced implementation of the gradient boosting algorithm designed for speed and performance. It has become one of the most popular algorithms in machine learning competitions and real-world applications due to its robust performance and efficiency.

Key Features of XGBoost:

1. **Regularization**: XGBoost includes regularization parameters (L1 and L2) to control overfitting, which is not present in standard gradient boosting implementations.
2. **Parallel Processing**: It supports parallel processing, which speeds up the computation, especially for large datasets.
3. **Tree Pruning**: Uses a more sophisticated approach to prune trees, stopping the growth of branches when no further improvement is observed.
4. **Handling Missing Values**: XGBoost has a built-in mechanism to handle missing values, determining the best direction to handle missing data during training.
5. **Sparsity Awareness**: It efficiently handles sparse data and missing values using a sparsity-aware algorithm.
6. **Weighted Quantile Sketch**: This allows for better handling of weighted data points and large datasets.

How XGBoost Works:

1. **Initialization**: Start with an initial prediction (usually the mean of the target variable).
2. **Iterative Boosting**: In each iteration, a new decision tree is added to the model. This tree is trained to predict the residual errors of the existing model, effectively improving the overall prediction.
3. **Objective Function**: The algorithm minimizes a specific objective function, which is a combination of a loss function (measuring the difference between actual and predicted values) and a regularization term (to control the model complexity).
4. **Model Update**: The predictions from the new tree are combined with the existing model’s predictions, usually by a simple addition of the predictions (weighted by a learning rate).

Steps in XGBoost Algorithm:

1. **Compute Residuals**: Calculate the difference between the observed and predicted values (residuals).
2. **Fit Decision Tree**: Fit a decision tree to these residuals.
3. **Update Predictions**: Adjust the existing predictions by adding the new tree's predictions, scaled by a learning rate.
4. **Repeat**: Repeat the process for a predefined number of iterations or until the model converges.

Advantages:

- **Performance**: XGBoost often provides superior performance and accuracy compared to other algorithms.
- **Flexibility**: It can be used for both classification and regression tasks.
- **Efficiency**: Optimized for speed and performance, making it suitable for large datasets.
- **Interpretability**: Provides feature importance scores, helping in understanding the influence of features.

Disadvantages:

- **Complexity**: It is more complex to implement and tune compared to simpler algorithms.
- **Resource Intensive**: Can be computationally expensive, requiring significant memory and processing power.

It is mostly used in healthcare sector for the prediction of various harmful diseases.

Other Machine Learning Algorithms:

1. Linear Regression

- **Purpose**: Used for predicting a continuous target variable.
- **Mechanism**: It models the relationship between the target variable and one or more predictor variables using a linear equation.

2. Logistic Regression

- **Purpose**: Used for binary classification problems.
- **Mechanism**: It models the probability of a binary outcome using a logistic function.

3. Support Vector Machines (SVM)

- **Purpose**: Used for classification and regression tasks.
- **Mechanism**: It finds the hyperplane that best separates the data into classes, maximizing the margin between the closest points of the classes (support vectors).

4. k-Nearest Neighbors (k-NN)

- **Purpose**: Used for classification and regression.
- **Mechanism**: It classifies a data point based on the majority class among its k-nearest neighbors.

5. Naive Bayes

- **Purpose**: Used for classification problems.
- **Mechanism**: It applies Bayes' theorem with the assumption that features are independent, computing the probability of each class and selecting the highest one.

6. Decision Trees

- **Purpose**: Used for classification and regression.
- **Mechanism**: It splits the data into subsets based on the value of input features, creating a tree structure where each node represents a decision rule.

7. Neural Networks

- **Purpose**: Used for a wide range of tasks including classification, regression, and more complex tasks like image and speech recognition.

- **Mechanism**: It consists of layers of interconnected nodes (neurons), where each connection has a weight, and data is passed through these layers with activations and transformations to produce the output.

8. Principal Component Analysis (PCA)

- **Purpose**: Used for dimensionality reduction.
- **Mechanism**: It transforms data to a new coordinate system where the greatest variances come to lie on the first coordinates (principal components).

9. Gradient Boosting Machines (GBM)

- **Purpose**: Used for classification and regression.
- **Mechanism**: It builds an ensemble of trees in a stage-wise fashion, where each tree corrects errors of the previous ones by focusing on the residuals.

These algorithms are fundamental tools in the machine learning toolkit, each used for different types of problems and data characteristics. Understanding their mechanisms, advantages, and limitations helps in selecting the right approach for a given task.

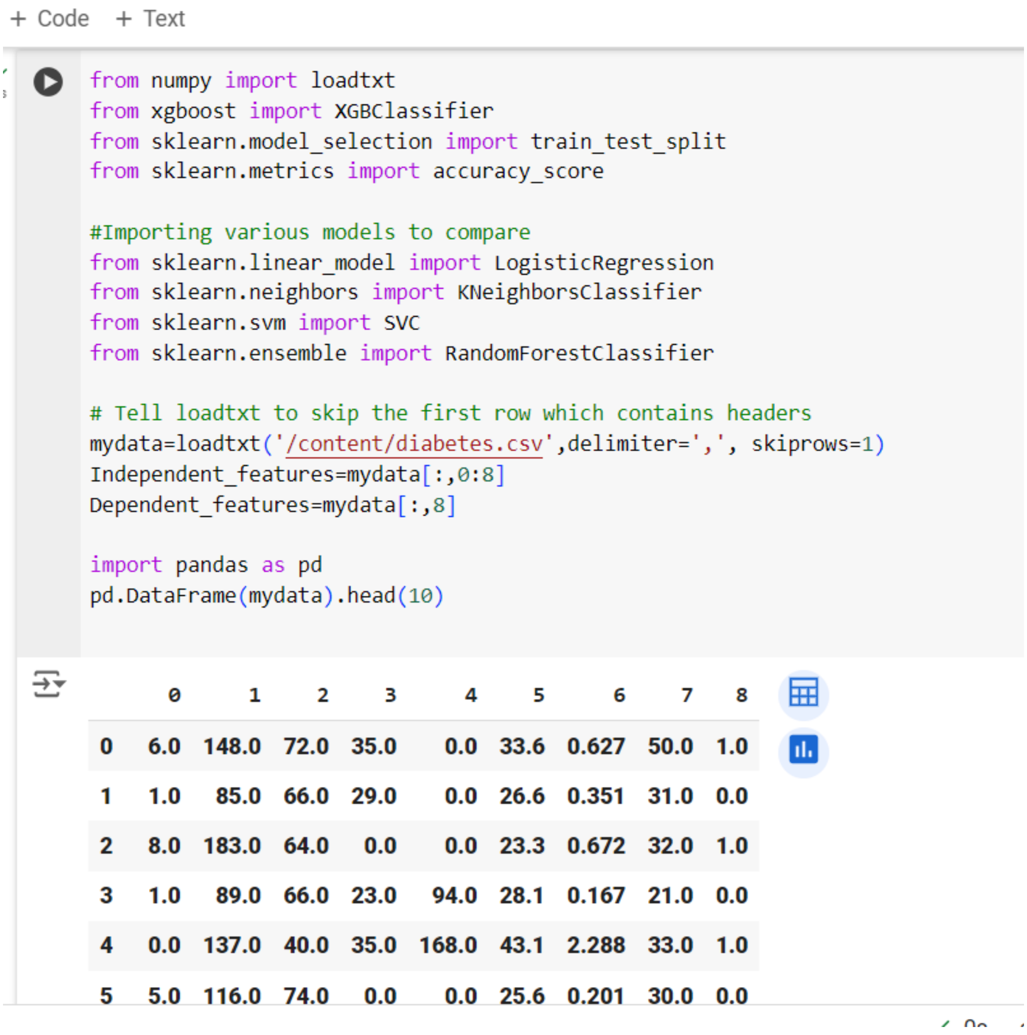
Best algorithm?

There is no single "best" machine learning algorithm that works universally for all problems. The effectiveness of an algorithm depends on various factors such as the nature of the data, the problem at hand, and the specific requirements of the task. Here are some key considerations to help determine the most suitable algorithm for different scenarios:

Factors to Consider:

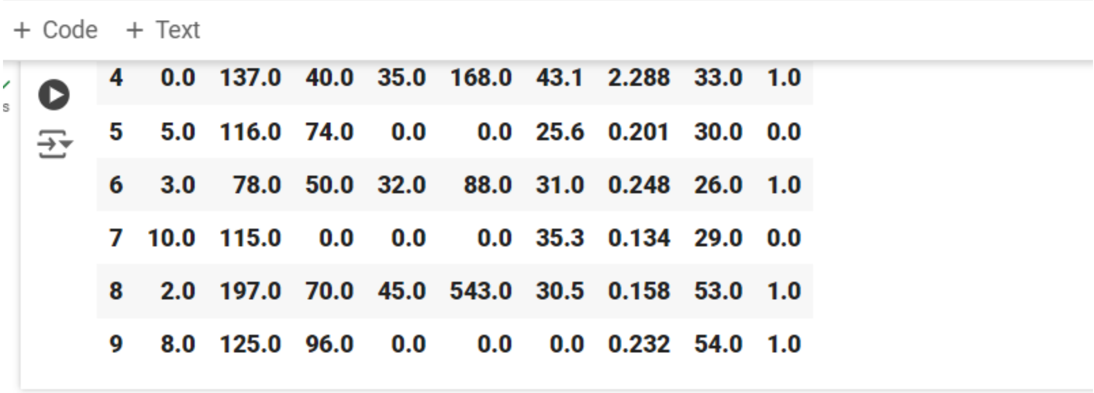
1. Type of Problem: Whether the problem is classification, regression, clustering, or a recommendation task.
2. Data Size and Quality: The volume of data available and the presence of missing or noisy data.
3. Computational Resources: The amount of computational power and memory available.
4. Interpretability: The need for understanding and explaining the model's decisions.
5. Performance Metrics: Specific performance criteria such as accuracy, precision, recall, F1-score, ROC-AUC, etc.

CODE:

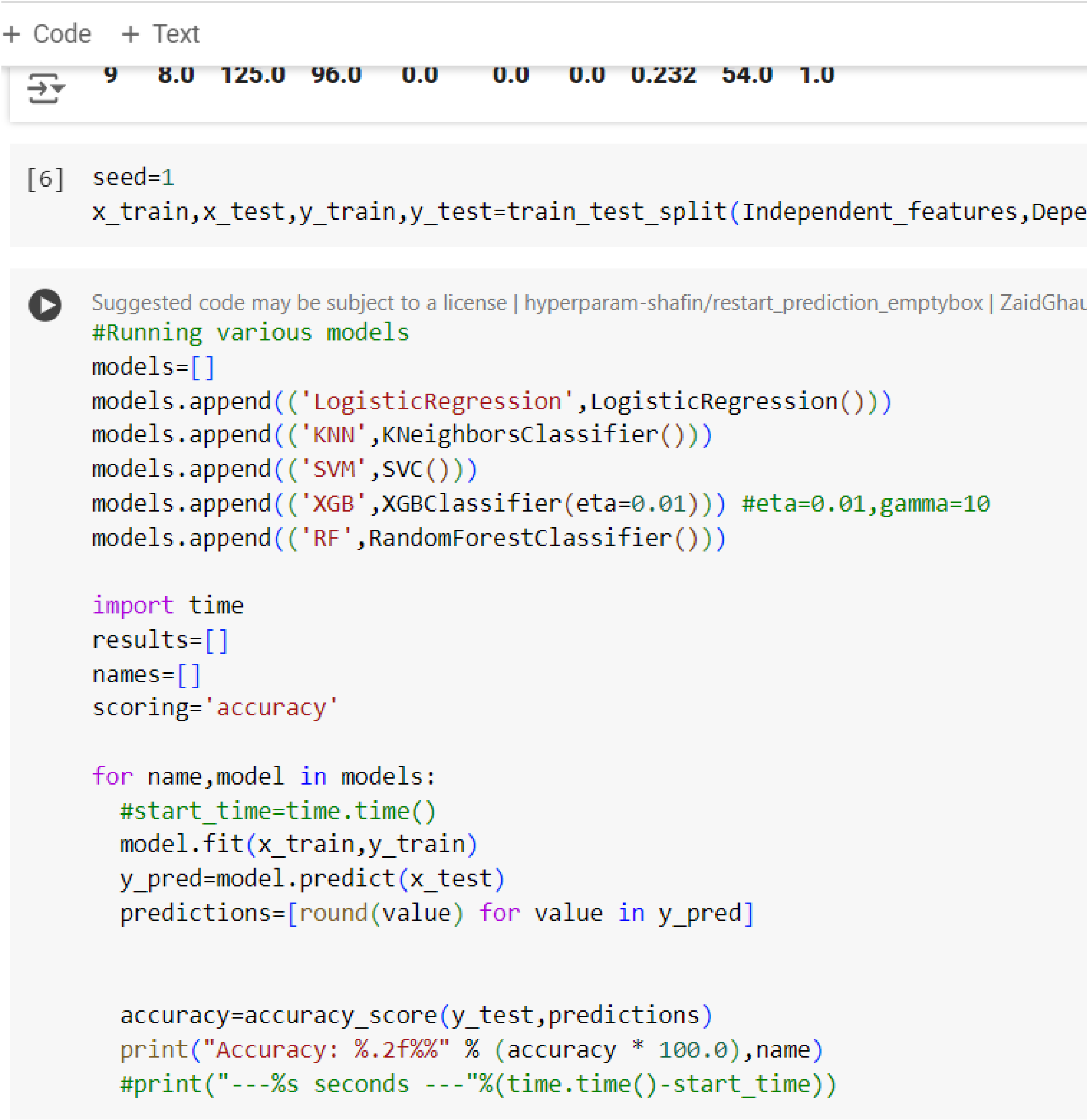


```
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#Importing various models to compare
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

# Tell loadtxt to skip the first row which contains headers
mydata=loadtxt('/content/diabetes.csv',delimiter=',', skiprows=1)
Independent_features=mydata[:,0:8]
Dependent_features=mydata[:,8]

import pandas as pd
pd.DataFrame(mydata).head(10)
```

```
     0      1     2     3      4     5      6     7    8
0  6.0  148.0  72.0  35.0    0.0  33.6  0.627  50.0  1.0
1  1.0   85.0  66.0  29.0    0.0  26.6  0.351  31.0  0.0
2  8.0  183.0  64.0   0.0    0.0  23.3  0.672  32.0  1.0
3  1.0   89.0  66.0  23.0   94.0  28.1  0.167  21.0  0.0
4  0.0  137.0  40.0  35.0  168.0  43.1  2.288  33.0  1.0
5  5.0  116.0  74.0   0.0    0.0  25.6  0.201  30.0  0.0
```



```
4   0.0  137.0  40.0  35.0  168.0  43.1  2.288  33.0  1.0
5   5.0  116.0  74.0   0.0    0.0  25.6  0.201  30.0  0.0
6   3.0   78.0  50.0  32.0   88.0  31.0  0.248  26.0  1.0
7  10.0  115.0   0.0   0.0    0.0  35.3  0.134  29.0  0.0
8   2.0  197.0  70.0  45.0  543.0  30.5  0.158  53.0  1.0
9   8.0  125.0  96.0   0.0    0.0   0.0  0.232  54.0  1.0
```



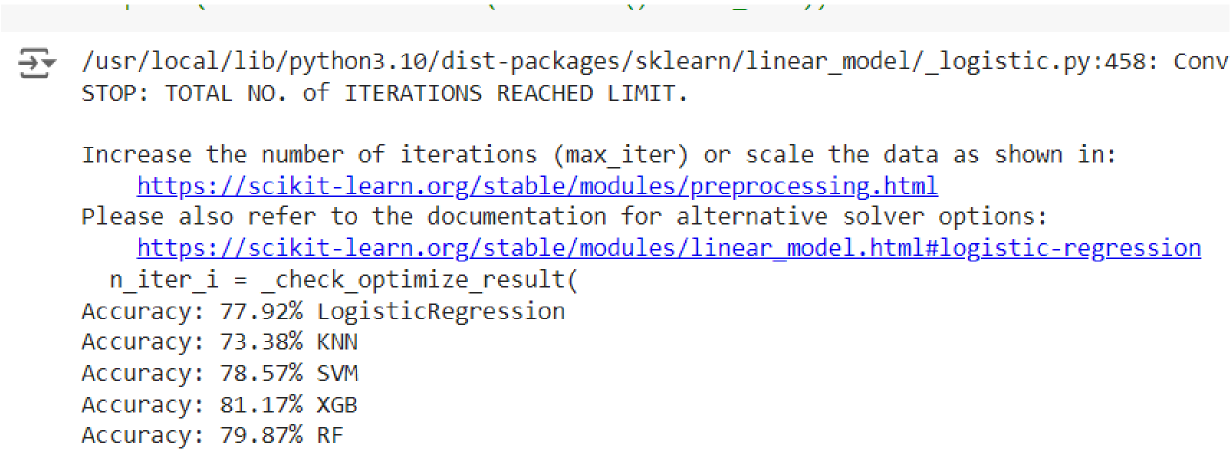
```
9   8.0  125.0  96.0   0.0    0.0   0.0  0.232  54.0  1.0
```

```
[6] seed=1
    x_train,x_test,y_train,y_test=train_test_split(Independent_features,Depe
```

Suggested code may be subject to a license | hyperparam-shafin/restart_prediction_emptybox | ZaidGhau

```
#Running various models
models=[]
models.append(('LogisticRegression',LogisticRegression()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('SVM',SVC()))
models.append(('XGB',XGBClassifier(eta=0.01))) #eta=0.01,gamma=10
models.append(('RF',RandomForestClassifier()))

import time
results=[]
names=[]
scoring='accuracy'

for name,model in models:
  #start_time=time.time()
  model.fit(x_train,y_train)
  y_pred=model.predict(x_test)
  predictions=[round(value) for value in y_pred]


  accuracy=accuracy_score(y_test,predictions)
  print("Accuracy: %.2f%%" % (accuracy * 100.0),name)
  #print("---%s seconds ---"%(time.time()-start_time))
```

OUTPUT:



```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Conv
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy: 77.92% LogisticRegression
Accuracy: 73.38% KNN
Accuracy: 78.57% SVM
Accuracy: 81.17% XGB
Accuracy: 79.87% RF
```

In the above code different algorithms have been tested on the dataset containing data of the diabetic patients.

The accuracy rate have been generated and according to the output the XGBoost algorithm has the maximum accuracy of 81.1% when value of eta was changed to 0.01. The accuracy further increases if we change the value of eta. Whereas the accuracy decreases if we increase the value of gamma as it will lead to pruning of the decision tree.

Accuracy of Random Forest algorithm was found to be a bit lower which is 79.87%

The lowest accuracy rate was found in KNeighbour algorithm of just 73.38%.

CREATING A MODEL FOR CANCER TYPE CLASSIFICATION (BREAST CANCER)

Tumors are classified into two main types**: benign and malignant**.

Benign tumors are non-cancerous growths that remain localized at their site of origin. They grow slowly and are usually encapsulated, which prevents them from invading surrounding tissues or spreading to other parts of the body. Though generally not life-threatening, benign tumors can still cause problems if they press on vital organs or structures.

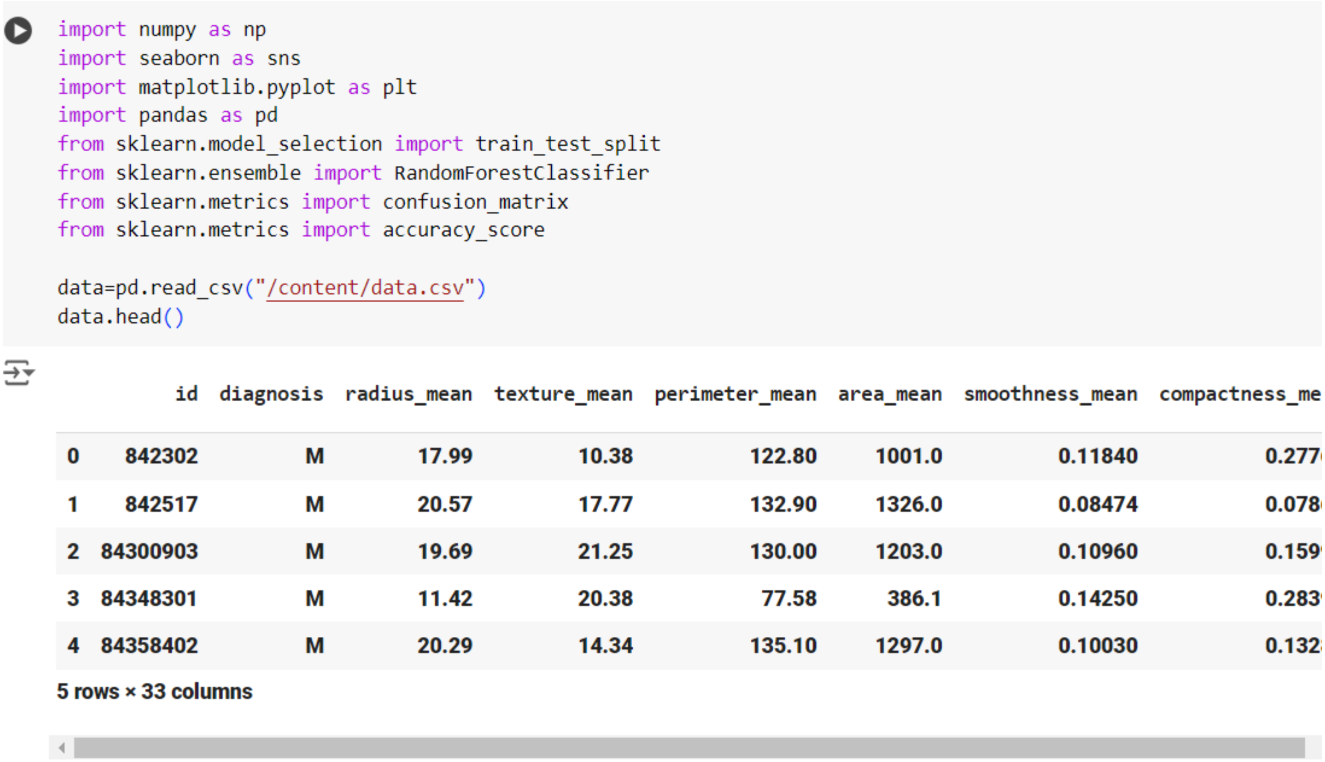
Malignant tumors, on the other hand, are cancerous and have the ability to invade nearby tissues and spread to distant parts of the body through a process known as metastasis. Malignant tumors are typically more aggressive, grow faster, and are often associated with a higher risk of recurrence after treatment.

The primary distinction between benign and malignant tumors lies in their potential to spread and cause harm to the body.

Malignant tumors are more harmful than benign tumors. This is because malignant tumors are cancerous and have the ability to invade nearby tissues and spread to other parts of the body through metastasis. Malignant tumors can lead to serious health complications and are associated with a higher risk of recurrence, making them more life-threatening compared to benign tumors, which typically remain localized and grow slowly without spreading.

MODEL AND ITS ACCURACY RATE:

The data imported from Kaggle contains 33 columns initially with details like id, diagnosis and features of the tumor.



```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

data=pd.read_csv("/content/data.csv")
data.head()
```

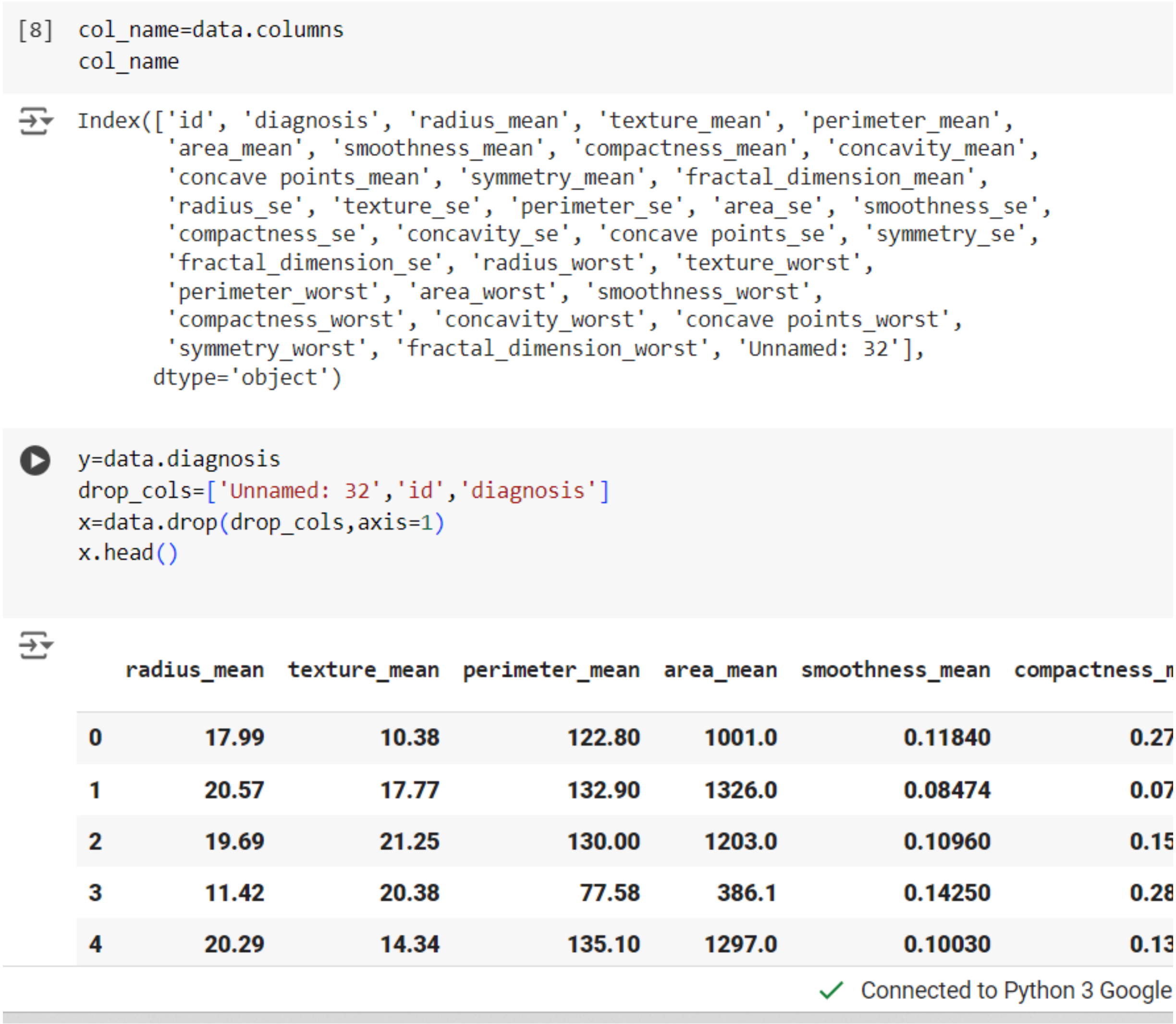
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.277
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.078
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.159
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.283
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.132

5 rows × 33 columns

-Importing necessary python libraries.

-Reading the data from the dataset.

-Displaying the data using head() function.



```
[8] col_name=data.columns
    col_name
```

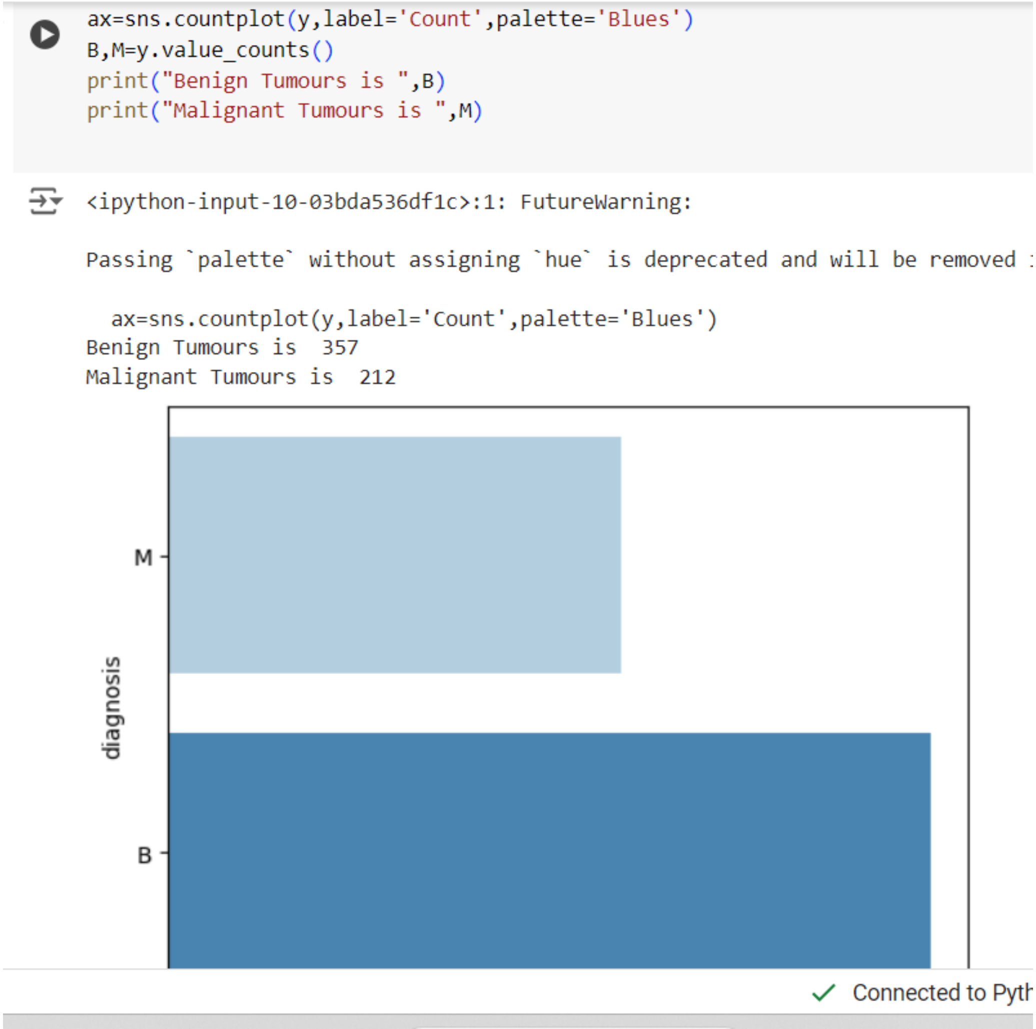
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

```
y=data.diagnosis
drop_cols=['Unnamed: 32','id','diagnosis']
x=data.drop(drop_cols,axis=1)
x.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
0	17.99	10.38	122.80	1001.0	0.11840	0.27
1	20.57	17.77	132.90	1326.0	0.08474	0.07
2	19.69	21.25	130.00	1203.0	0.10960	0.15
3	11.42	20.38	77.58	386.1	0.14250	0.28
4	20.29	14.34	135.10	1297.0	0.10030	0.13

-All column names have been displayed.

-Then dropped 3 columns and final data was displayed.

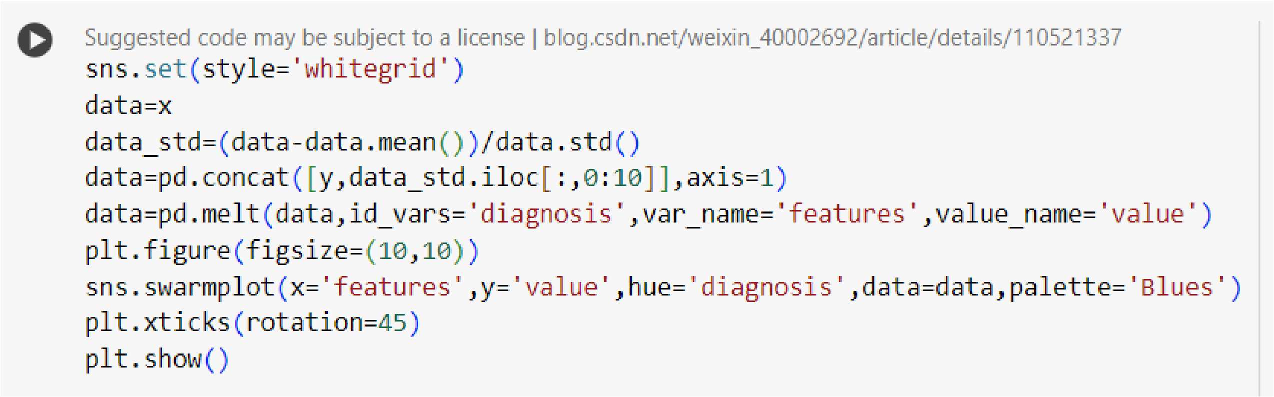


```
ax=sns.countplot(y,label='Count',palette='Blues')
B,M=y.value_counts()
print("Benign Tumours is ",B)
print("Malignant Tumours is ",M)
```

```
<ipython-input-10-03bda536df1c>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed :

  ax=sns.countplot(y,label='Count',palette='Blues')
Benign Tumours is  357
Malignant Tumours is  212
```

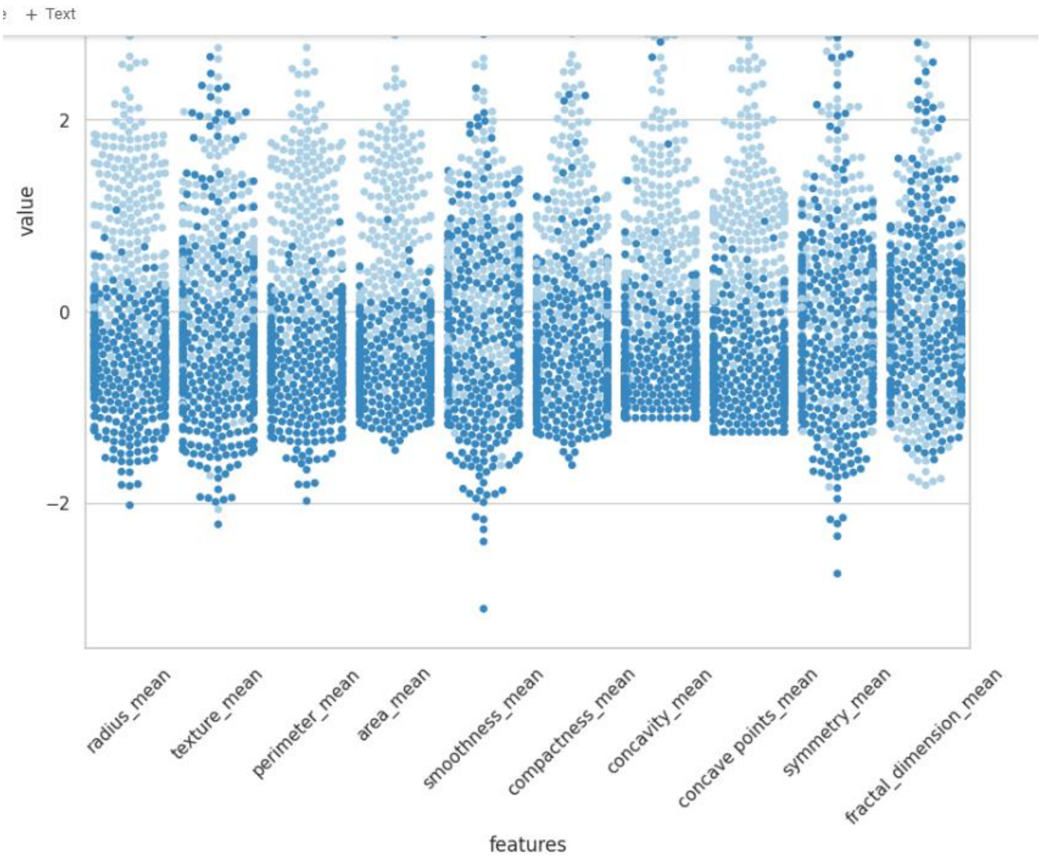
-The code uses Seaborn's countplot function to create a bar plot that visualizes the count of each category (benign and malignant) in the dataset y, with a blue colour palette.

- The code counts the occurrences of benign and malignant tumors in the dataset y, stores these counts in variables B and M, and prints the number of benign and malignant tumors.



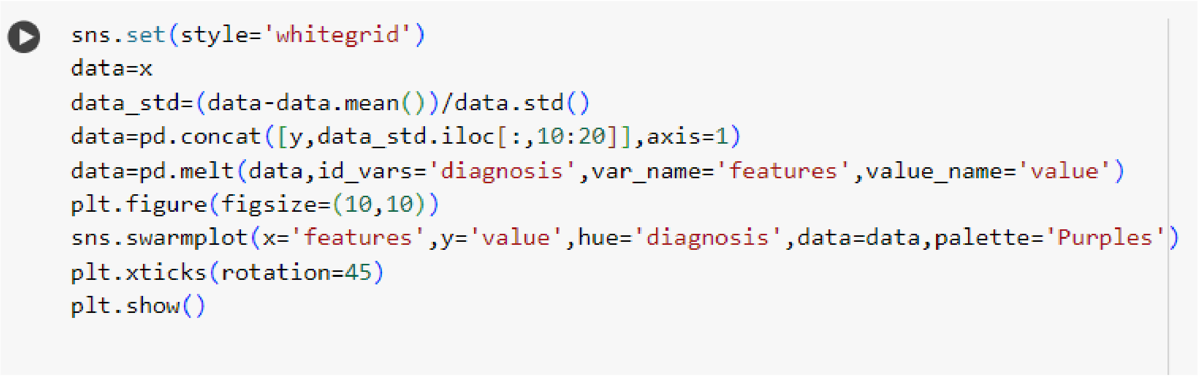
Suggested code may be subject to a license | blog.csdn.net/weixin_40002692/article/details/110521337

```
sns.set(style='whitegrid')
data=x
data_std=(data-data.mean())/data.std()
data=pd.concat([y,data_std.iloc[:,0:10]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data,palette='Blues')
plt.xticks(rotation=45)
plt.show()
```

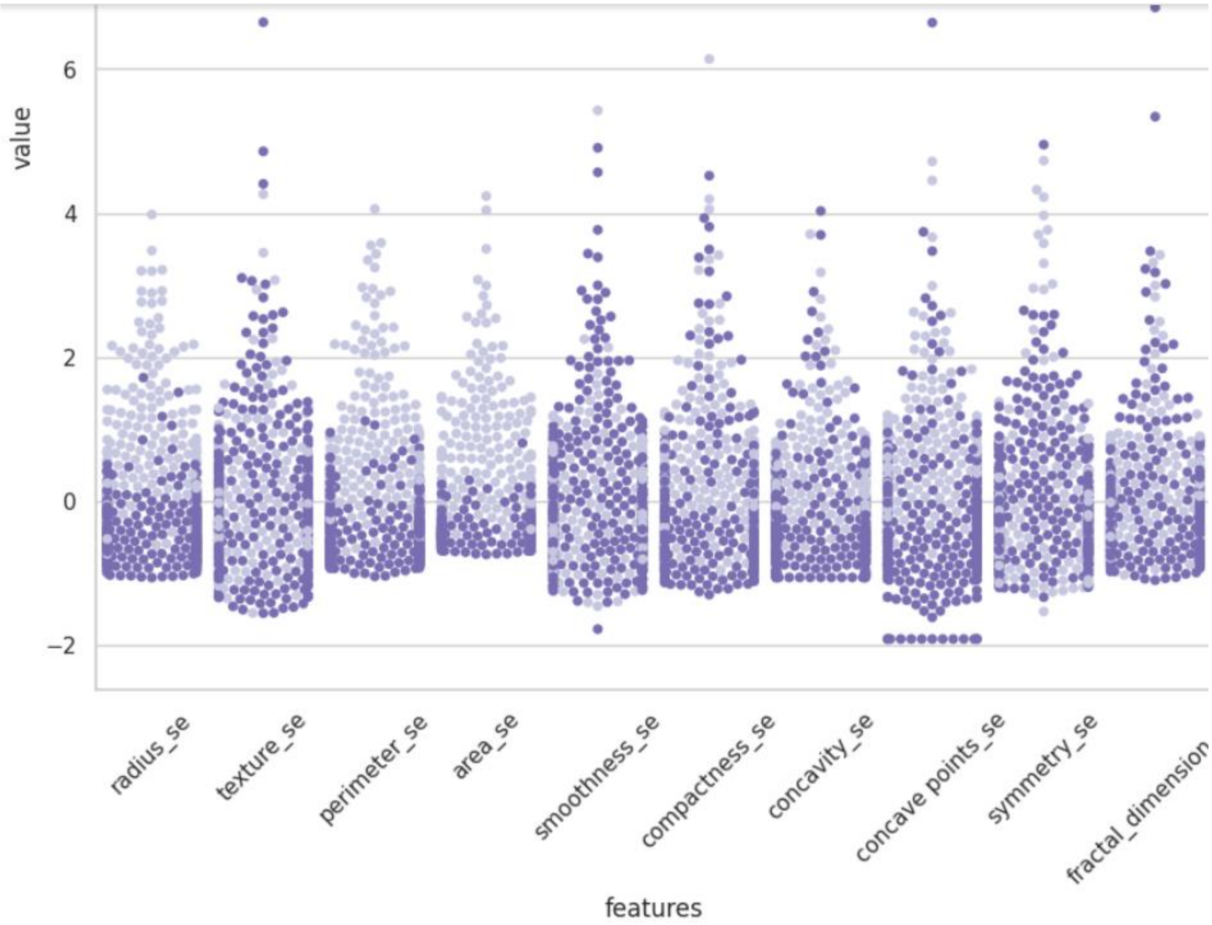


-**Standardization and Data Preparation**: The code standardizes the **first 10 columns** of the dataset x by subtracting the mean and dividing by the standard deviation. It then concatenates these standardized features with the y dataset (assumed to contain the 'diagnosis' label), and reshapes the data for visualization using the melt function.

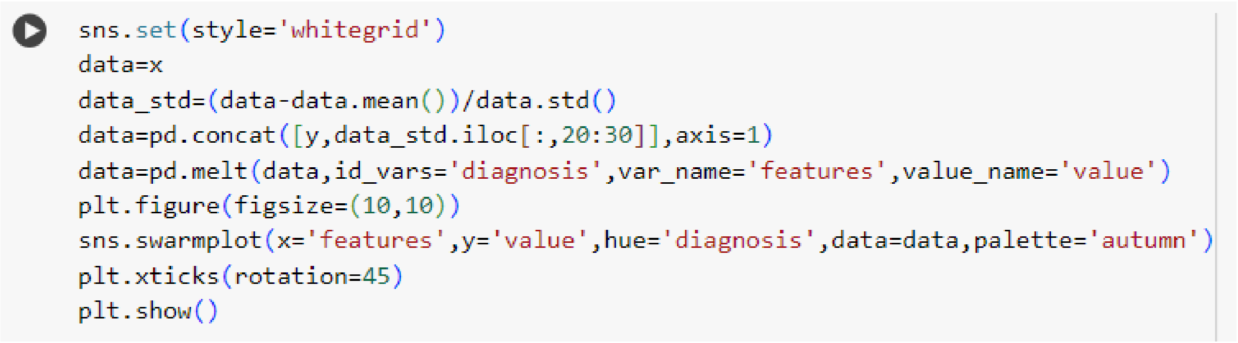
-**Swarm Plot Visualization**: It creates a swarm plot to visualize the distribution of the standardized feature values for benign and malignant tumors, differentiating the diagnoses with colours from the blue palette. The plot displays the feature names on the x-axis, values on the y-axis, and rotates the x-axis labels for better readability.



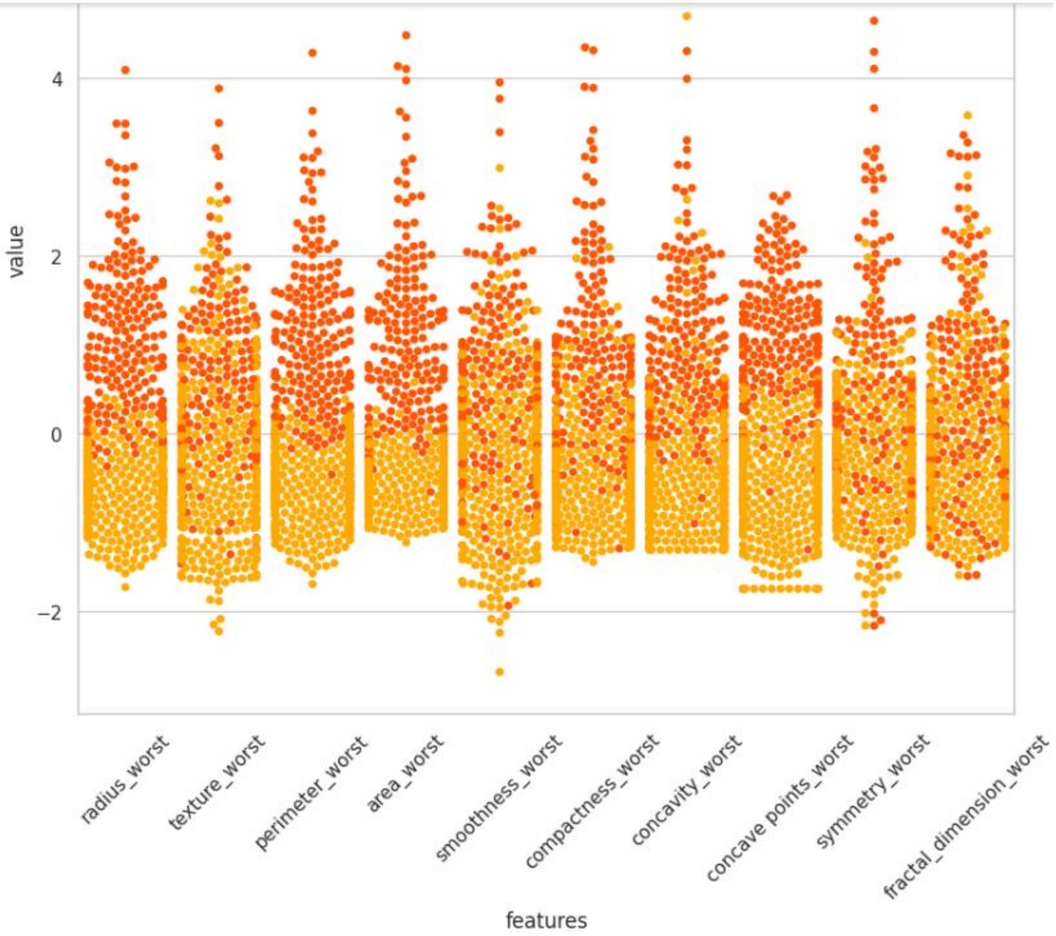
```
sns.set(style='whitegrid')
data=x
data_std=(data-data.mean())/data.std()
data=pd.concat([y,data_std.iloc[:,10:20]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data,palette='Purples')
plt.xticks(rotation=45)
plt.show()
```



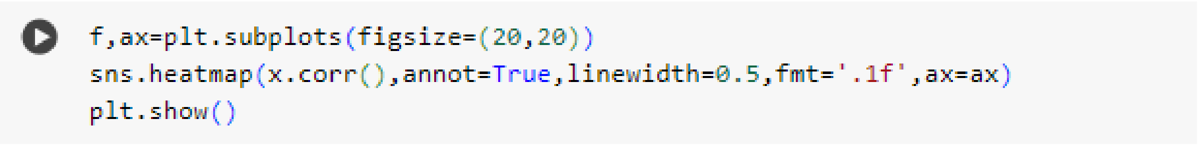
-The next 10 columns have used the Purple palette to plot the swarmplot.



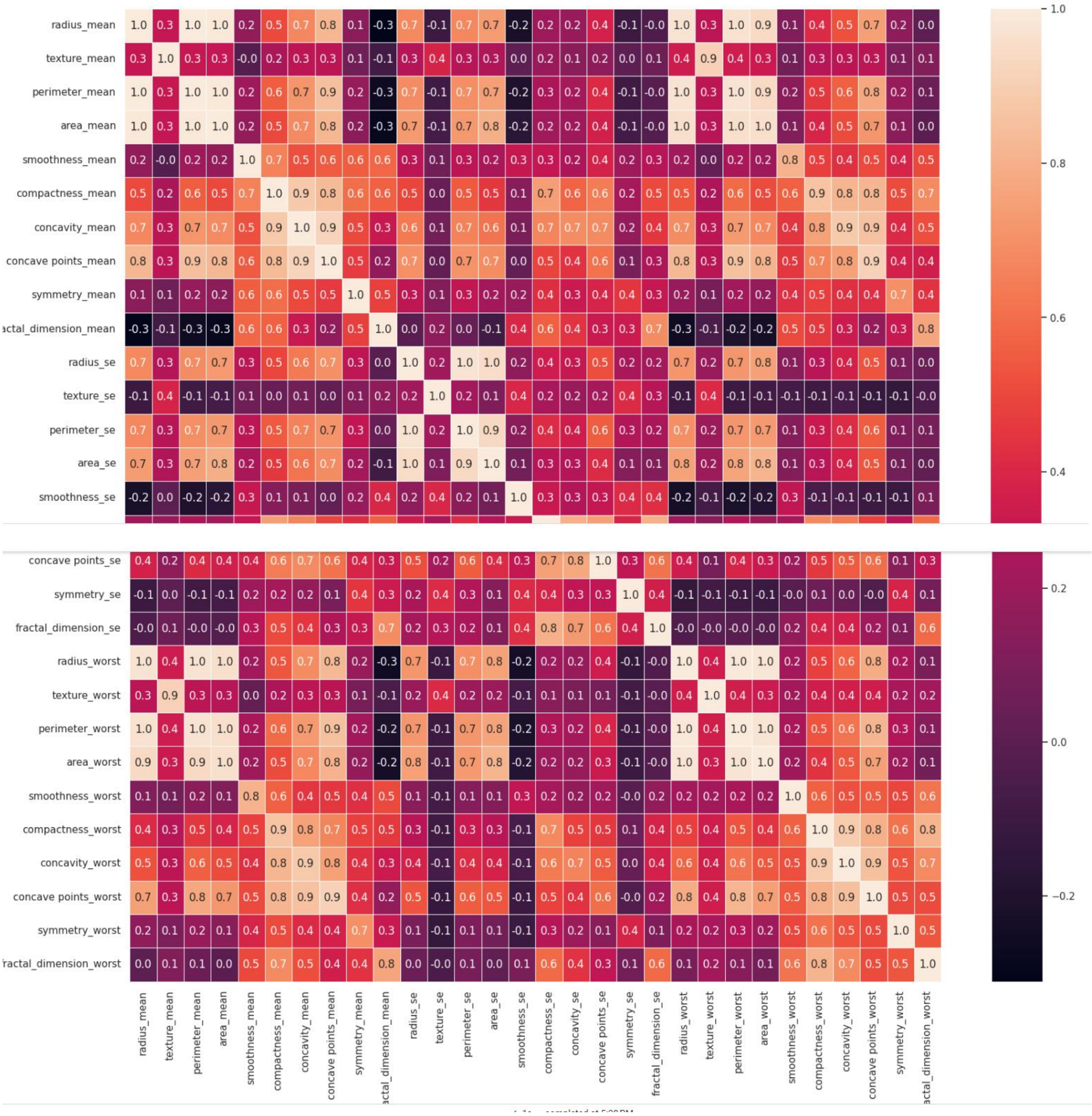
```
sns.set(style='whitegrid')
data=x
data_std=(data-data.mean())/data.std()
data=pd.concat([y,data_std.iloc[:,20:30]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data,palette='autumn')
plt.xticks(rotation=45)
plt.show()
```



-The last ten columns have used the ‘autumn’ palette for the swarmplot as shown above.

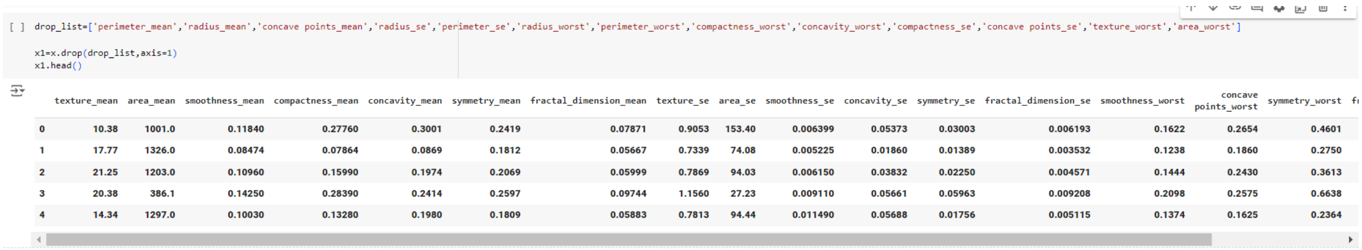


```
f,ax=plt.subplots(figsize=(20,20))
sns.heatmap(x.corr(),annot=True,linewidth=0.5,fmt='.1f',ax=ax)
plt.show()
```



-**Correlation Heatmap Creation**: The code generates a heatmap using Seaborn to visualize the correlation matrix of the dataset x, which shows the pairwise correlation coefficients between the features in x.

-**Enhanced Visualization**: The heatmap includes annotations for each cell displaying the correlation values with one decimal precision, and it is displayed with a specified figure size of 20x20 inches for clarity. The plot is shown with grid lines (linewidth=0.5) to distinguish between cells.

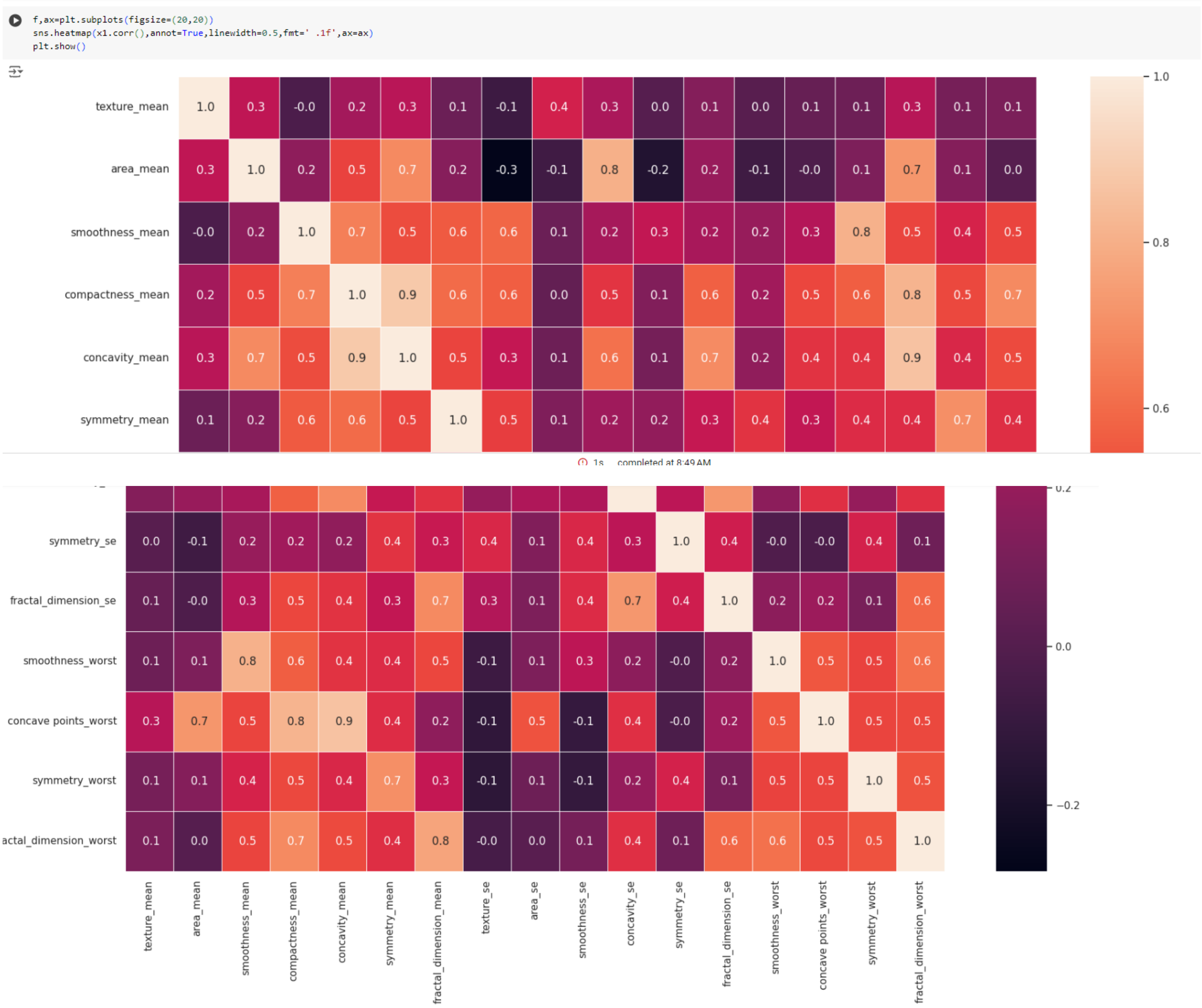


```
[ ] drop_list=['perimeter_mean','radius_mean','concave points_mean','radius_se','perimeter_se','radius_worst','perimeter_worst','compactness_worst','concavity_worst','compactness_se','concave points_se','texture_worst','area_worst']

    x1=x.drop(drop_list,axis=1)
    x1.head()
```

	texture_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	symmetry_mean	fractal_dimension_mean	texture_se	area_se	smoothness_se	concavity_se	symmetry_se	fractal_dimension_se	smoothness_worst	concave points_worst	symmetry_worst	fr
0	10.38	1001.0	0.11840	0.27760	0.3001	0.2419	0.07871	0.9053	153.40	0.006399	0.05373	0.03003	0.006193	0.1622	0.2654	0.4601	
1	17.77	1326.0	0.08474	0.07864	0.0869	0.1812	0.05667	0.7339	74.08	0.005225	0.01860	0.01389	0.003532	0.1238	0.1860	0.2750	
2	21.25	1203.0	0.10960	0.15990	0.1974	0.2069	0.05999	0.7869	94.03	0.006150	0.03832	0.02250	0.004571	0.1444	0.2430	0.3613	
3	20.38	386.1	0.14250	0.28390	0.2414	0.2597	0.09744	1.1560	27.23	0.009110	0.05661	0.05963	0.009208	0.2098	0.2575	0.6638	
4	14.34	1297.0	0.10030	0.13280	0.1980	0.1809	0.05883	0.7813	94.44	0.011490	0.05688	0.01756	0.005115	0.1374	0.1625	0.2364	

-**Feature Dropping**: The code defines a list of feature names (drop_list) to be removed from the dataset x. These features are related to measurements of perimeter, radius, concavity, compactness, texture, and area.

-**Creating a New Dataset**: It creates a new dataset x1 by dropping the specified features from x and then displays the first few rows of the updated dataset using x1.head().

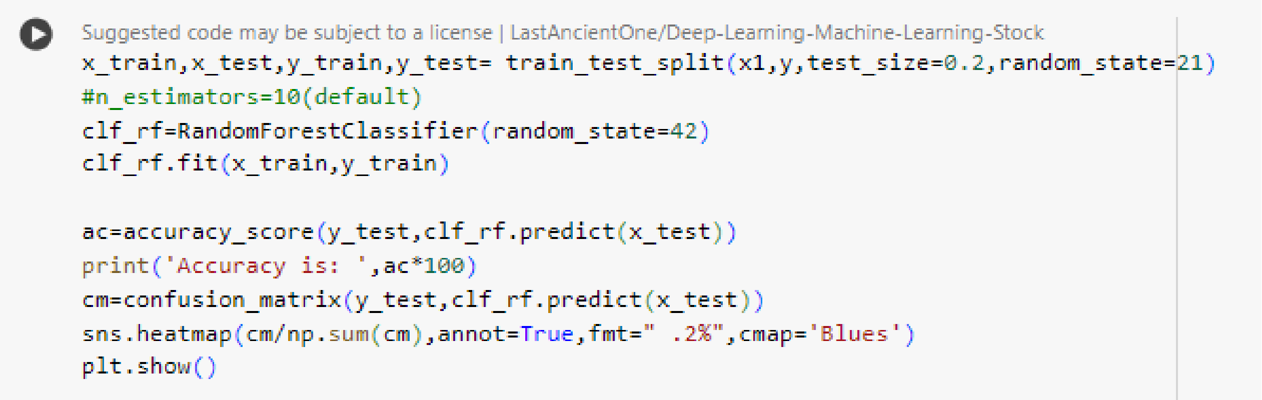


```
f,ax=plt.subplots(figsize=(20,20))
sns.heatmap(x1.corr(),annot=True,linewidth=0.5,fmt=' .1f',ax=ax)
plt.show()
```

1s completed at 8:49 AM

texture_mean 1.0 0.3 -0.0 0.2 0.3 0.1 -0.1 0.4 0.3 0.0 0.1 0.0 0.1 0.1 0.3 0.1 0.1
area_mean 0.3 1.0 0.2 0.5 0.7 0.2 -0.3 -0.1 0.8 -0.2 0.2 -0.1 -0.0 0.1 0.7 0.1 0.0
smoothness_mean -0.0 0.2 1.0 0.7 0.5 0.6 0.6 0.1 0.2 0.3 0.2 0.2 0.3 0.8 0.5 0.4 0.5
compactness_mean 0.2 0.5 0.7 1.0 0.9 0.6 0.6 0.0 0.5 0.1 0.6 0.2 0.5 0.6 0.8 0.5 0.7
concavity_mean 0.3 0.7 0.5 0.9 1.0 0.5 0.3 0.1 0.6 0.1 0.7 0.2 0.4 0.4 0.9 0.4 0.5
symmetry_mean 0.1 0.2 0.6 0.6 0.5 1.0 0.5 0.1 0.2 0.2 0.3 0.4 0.3 0.4 0.4 0.7 0.4
1.0
0.8
0.6
symmetry_se 0.0 -0.1 0.2 0.2 0.2 0.4 0.3 0.4 0.1 0.4 0.3 1.0 0.4 -0.0 -0.0 0.4 0.1
fractal_dimension_se 0.1 -0.0 0.3 0.5 0.4 0.3 0.7 0.3 0.1 0.4 0.7 0.4 1.0 0.2 0.2 0.1 0.6
smoothness_worst 0.1 0.1 0.8 0.6 0.4 0.4 0.5 -0.1 0.1 0.3 0.2 -0.0 0.2 1.0 0.5 0.5 0.6
concave points_worst 0.3 0.7 0.5 0.8 0.9 0.4 0.2 -0.1 0.5 -0.1 0.4 -0.0 0.2 0.5 1.0 0.5 0.5
symmetry_worst 0.1 0.1 0.4 0.5 0.4 0.7 0.3 -0.1 0.1 -0.1 0.2 0.4 0.1 0.5 0.5 1.0 0.5
actal_dimension_worst 0.1 0.0 0.5 0.7 0.5 0.4 0.8 -0.0 0.0 0.1 0.4 0.1 0.6 0.6 0.5 0.5 1.0
0.2
0.0
−0.2
texture_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
symmetry_mean
fractal_dimension_mean
texture_se
area_se
smoothness_se
concavity_se
symmetry_se
fractal_dimension_se
smoothness_worst
concave points_worst
symmetry_worst
fractal_dimension_worst

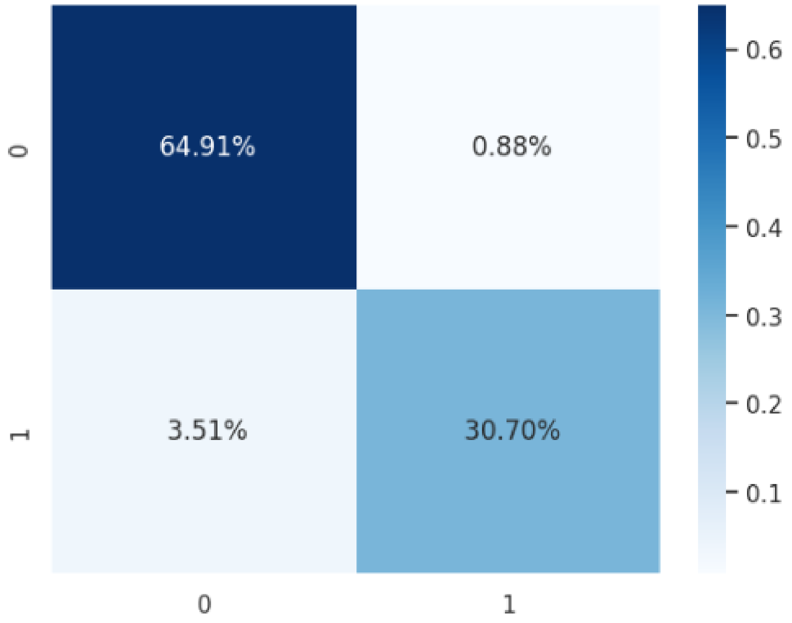
-The new plot above does not have the redundant cells.



Suggested code may be subject to a license | LastAncientOne/Deep-Learning-Machine-Learning-Stock

```
x_train,x_test,y_train,y_test= train_test_split(x1,y,test_size=0.2,random_state=21)
#n_estimators=10(default)
clf_rf=RandomForestClassifier(random_state=42)
clf_rf.fit(x_train,y_train)

ac=accuracy_score(y_test,clf_rf.predict(x_test))
print('Accuracy is: ',ac*100)
cm=confusion_matrix(y_test,clf_rf.predict(x_test))
sns.heatmap(cm/np.sum(cm),annot=True,fmt=" .2%",cmap='Blues')
plt.show()
```

```
Accuracy is:  95.6140350877193
```



-The accuracy of the model came out to be 95.61%.

1. **Data Splitting:**

The code splits the dataset `x1` and the labels `y` into training and testing sets. 80% of the data is used for training (`x_train`, `y_train`), and 20% is used for testing (`x_test`, `y_test`). The `random_state=21` ensures reproducibility.

2)Model Training:

It initializes a RandomForestClassifier with a `random_state=42` for reproducibility and fits the model using the training data (`x_train`, `y_train`).

3. **Model Evaluation**:

- It calculates the accuracy of the model by predicting the labels for the test data (`x_test`) and comparing them to the true labels (`y_test`). The accuracy score is printed as a percentage.

- It generates a confusion matrix from the predictions and the true labels to assess the model's performance in terms of true positives, true negatives, false positives, and false negatives. This matrix is normalized by the sum of all its elements to display percentages.

4**. Visualization**: The confusion matrix is visualized using a heatmap with annotations, showing the percentage values in a blue color map.

Result Obtained:

The result is the printed accuracy of the RandomForestClassifier on the test set and a heatmap of the normalized confusion matrix, which provides a visual representation of the model's performance in classifying the test data. The heatmap shows how well the model distinguishes between the different classes (e.g., benign and malignant tumors).

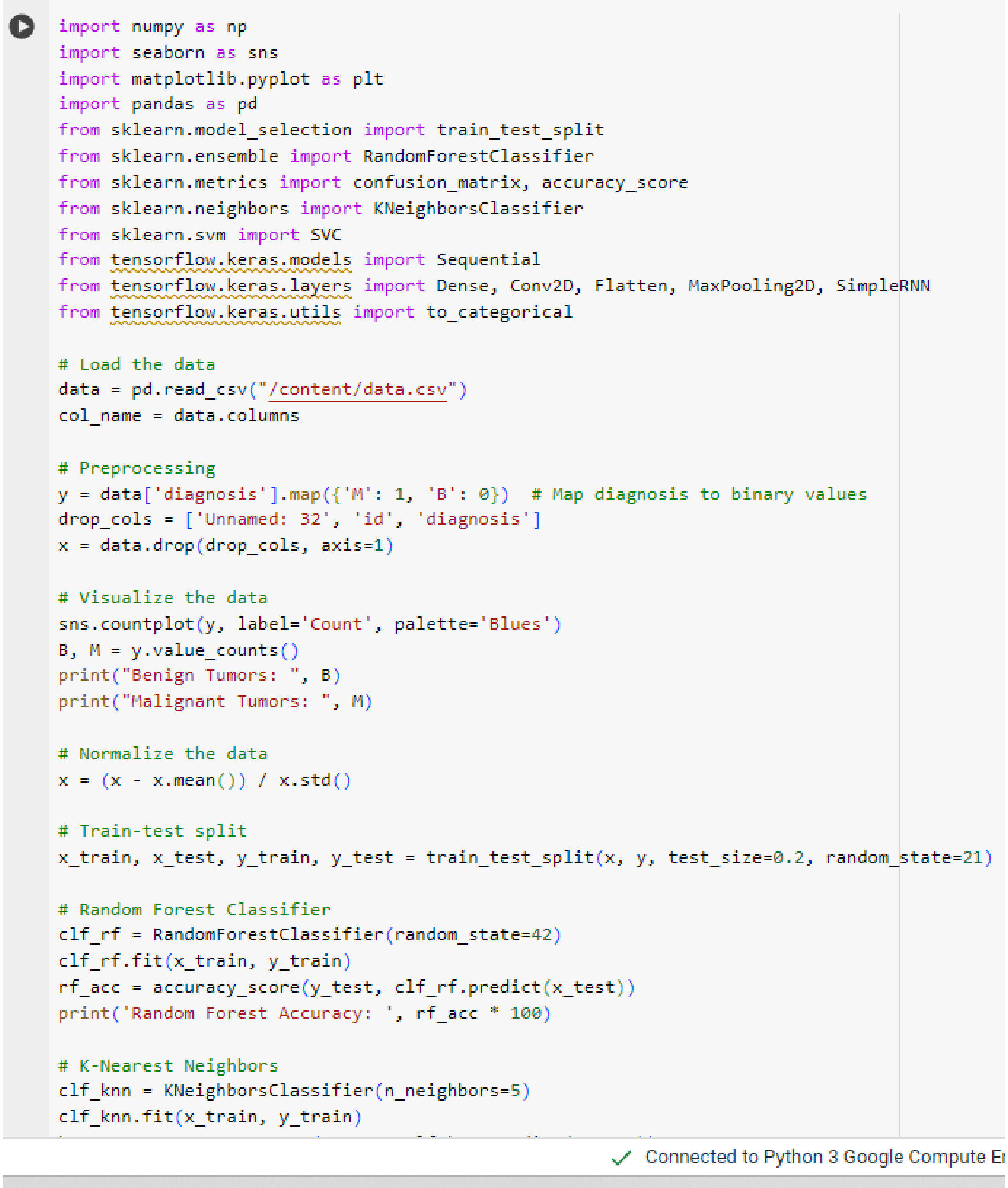
In the course of this research, I undertook a comprehensive study to explore the efficacy of various AI, ML, and DL algorithms in the detection of breast cancer by working on a research paper. My approach involved leveraging established machine learning techniques such as Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM), alongside advanced deep learning models including Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). By training and fine-tuning these models, I was able to assess and compare their performance, ultimately demonstrating the strengths and potential of these algorithms in accurately identifying malignant and benign cases. This rigorous analysis not only highlighted the superior performance of deep learning models but also underscored the practical challenges and considerations necessary for implementing AI in clinical settings.

AI and ML algorithms can analyze large amounts of medical data, such as images, genetic information, and clinical records, to detect patterns that may indicate cancer. For example, DL algorithms have shown great ability in analyzing medical images and have even outperformed human radiologists in detecting certain types of cancerous lesions. These technologies can also improve the predictive power of diagnostic tools, allowing for earlier and more accurate diagnoses. Additionally, AI-driven models can help create personalized treatment plans by analyzing the genetic and molecular profiles of tumors, leading to more targeted and effective therapies.

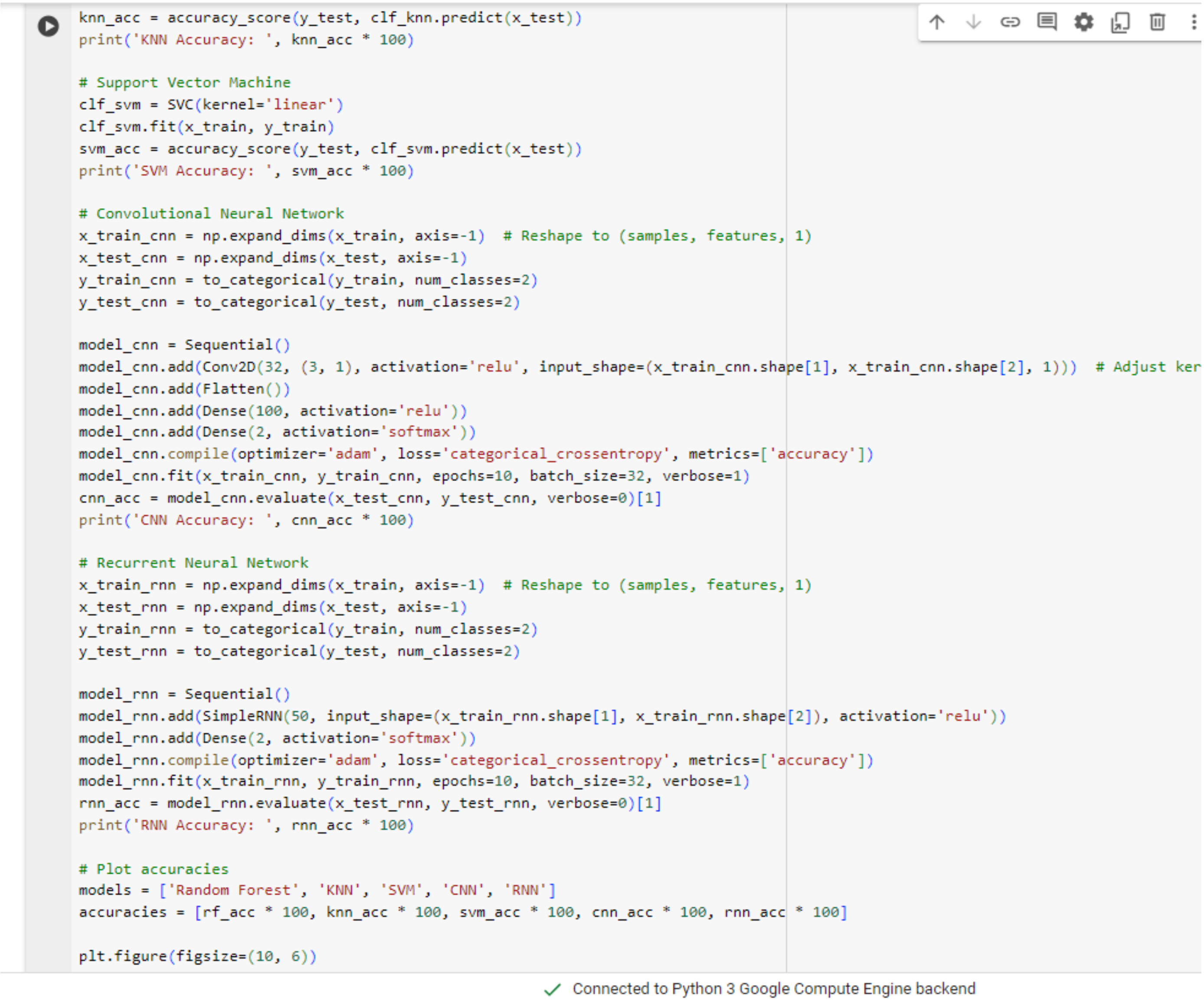
Several machine learning and deep learning algorithms have been employed to detect breast cancer using a given dataset. These algorithms include Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). Each of these algorithms approaches the problem differently, leveraging unique techniques and strengths to achieve the goal of accurate cancer detection.

DATASET: The dataset used in this study is the Breast Cancer Wisconsin (Diagnostic) dataset, which is widely recognized and used for breast cancer detection research. The dataset comprises 569 instances of breast cancer cases. These features capture various characteristics of the cell nuclei present in the image, such as radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.

CODE:

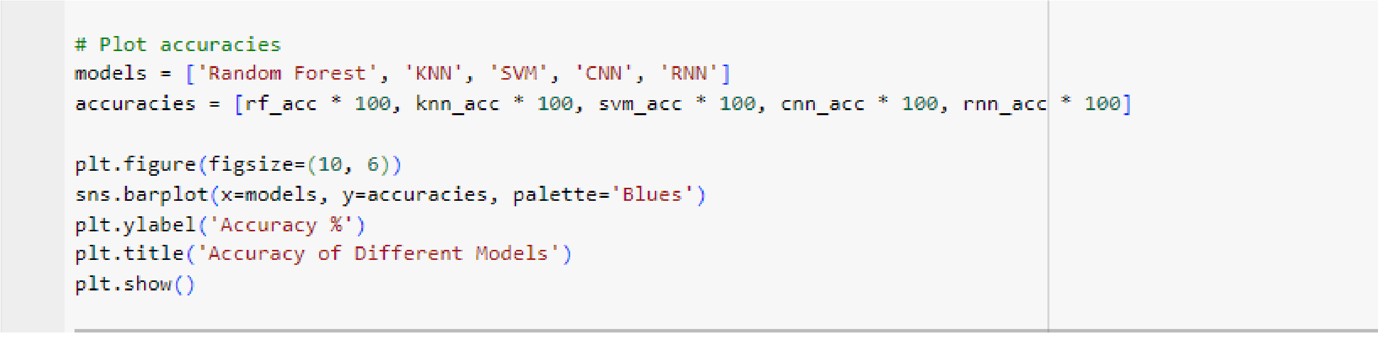


```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, SimpleRNN
from tensorflow.keras.utils import to_categorical

# Load the data
data = pd.read_csv("/content/data.csv")
col_name = data.columns

# Preprocessing
y = data['diagnosis'].map({'M': 1, 'B': 0})  # Map diagnosis to binary values
drop_cols = ['Unnamed: 32', 'id', 'diagnosis']
x = data.drop(drop_cols, axis=1)

# Visualize the data
sns.countplot(y, label='Count', palette='Blues')
B, M = y.value_counts()
print("Benign Tumors: ", B)
print("Malignant Tumors: ", M)

# Normalize the data
x = (x - x.mean()) / x.std()

# Train-test split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=21)

# Random Forest Classifier
clf_rf = RandomForestClassifier(random_state=42)
clf_rf.fit(x_train, y_train)
rf_acc = accuracy_score(y_test, clf_rf.predict(x_test))
print('Random Forest Accuracy: ', rf_acc * 100)

# K-Nearest Neighbors
clf_knn = KNeighborsClassifier(n_neighbors=5)
clf_knn.fit(x_train, y_train)
```



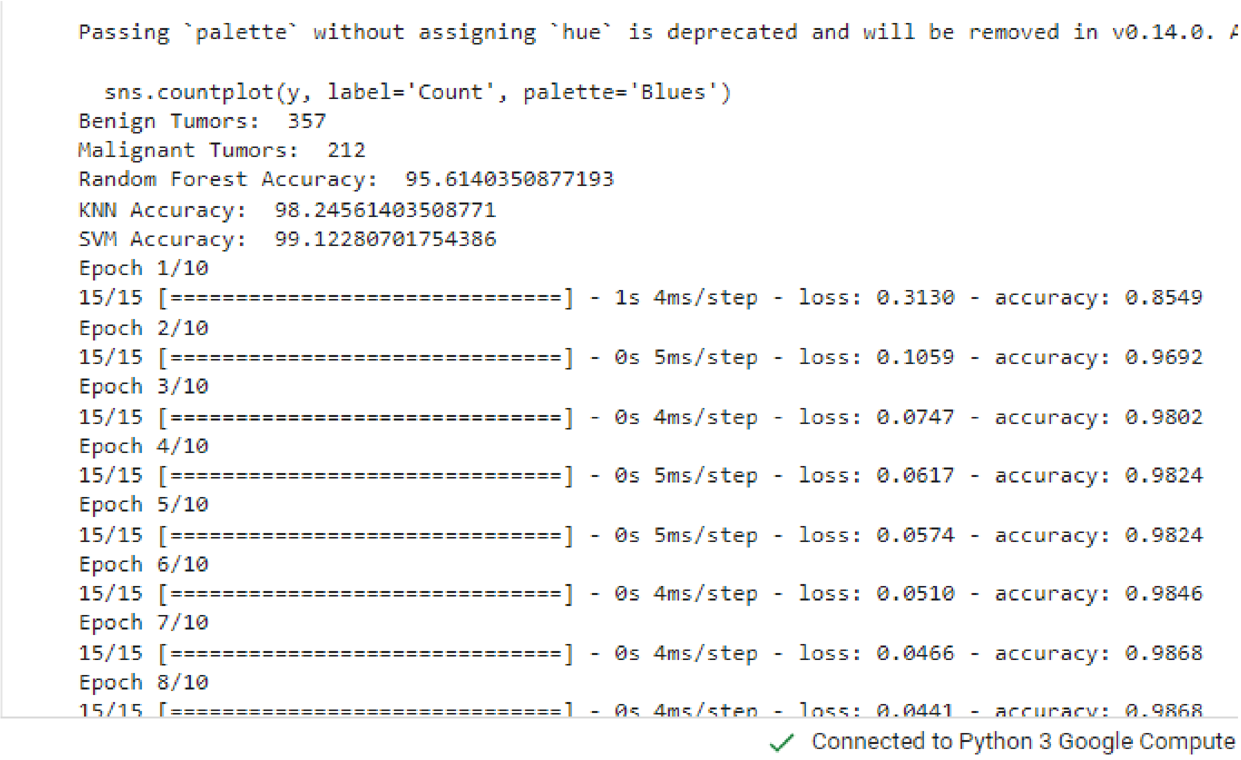
```
knn_acc = accuracy_score(y_test, clf_knn.predict(x_test))
print('KNN Accuracy: ', knn_acc * 100)

# Support Vector Machine
clf_svm = SVC(kernel='linear')
clf_svm.fit(x_train, y_train)
svm_acc = accuracy_score(y_test, clf_svm.predict(x_test))
print('SVM Accuracy: ', svm_acc * 100)

# Convolutional Neural Network
x_train_cnn = np.expand_dims(x_train, axis=-1)  # Reshape to (samples, features, 1)
x_test_cnn = np.expand_dims(x_test, axis=-1)
y_train_cnn = to_categorical(y_train, num_classes=2)
y_test_cnn = to_categorical(y_test, num_classes=2)

model_cnn = Sequential()
model_cnn.add(Conv2D(32, (3, 1), activation='relu', input_shape=(x_train_cnn.shape[1], x_train_cnn.shape[2], 1)))  # Adjust ker
model_cnn.add(Flatten())
model_cnn.add(Dense(100, activation='relu'))
model_cnn.add(Dense(2, activation='softmax'))
model_cnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_cnn.fit(x_train_cnn, y_train_cnn, epochs=10, batch_size=32, verbose=1)
cnn_acc = model_cnn.evaluate(x_test_cnn, y_test_cnn, verbose=0)[1]
print('CNN Accuracy: ', cnn_acc * 100)

# Recurrent Neural Network
x_train_rnn = np.expand_dims(x_train, axis=-1)  # Reshape to (samples, features, 1)
x_test_rnn = np.expand_dims(x_test, axis=-1)
y_train_rnn = to_categorical(y_train, num_classes=2)
y_test_rnn = to_categorical(y_test, num_classes=2)

model_rnn = Sequential()
model_rnn.add(SimpleRNN(50, input_shape=(x_train_rnn.shape[1], x_train_rnn.shape[2]), activation='relu'))
model_rnn.add(Dense(2, activation='softmax'))
model_rnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_rnn.fit(x_train_rnn, y_train_rnn, epochs=10, batch_size=32, verbose=1)
rnn_acc = model_rnn.evaluate(x_test_rnn, y_test_rnn, verbose=0)[1]
print('RNN Accuracy: ', rnn_acc * 100)

# Plot accuracies
models = ['Random Forest', 'KNN', 'SVM', 'CNN', 'RNN']
accuracies = [rf_acc * 100, knn_acc * 100, svm_acc * 100, cnn_acc * 100, rnn_acc * 100]

plt.figure(figsize=(10, 6))
```

✓ Connected to Python 3 Google Compute Engine backend

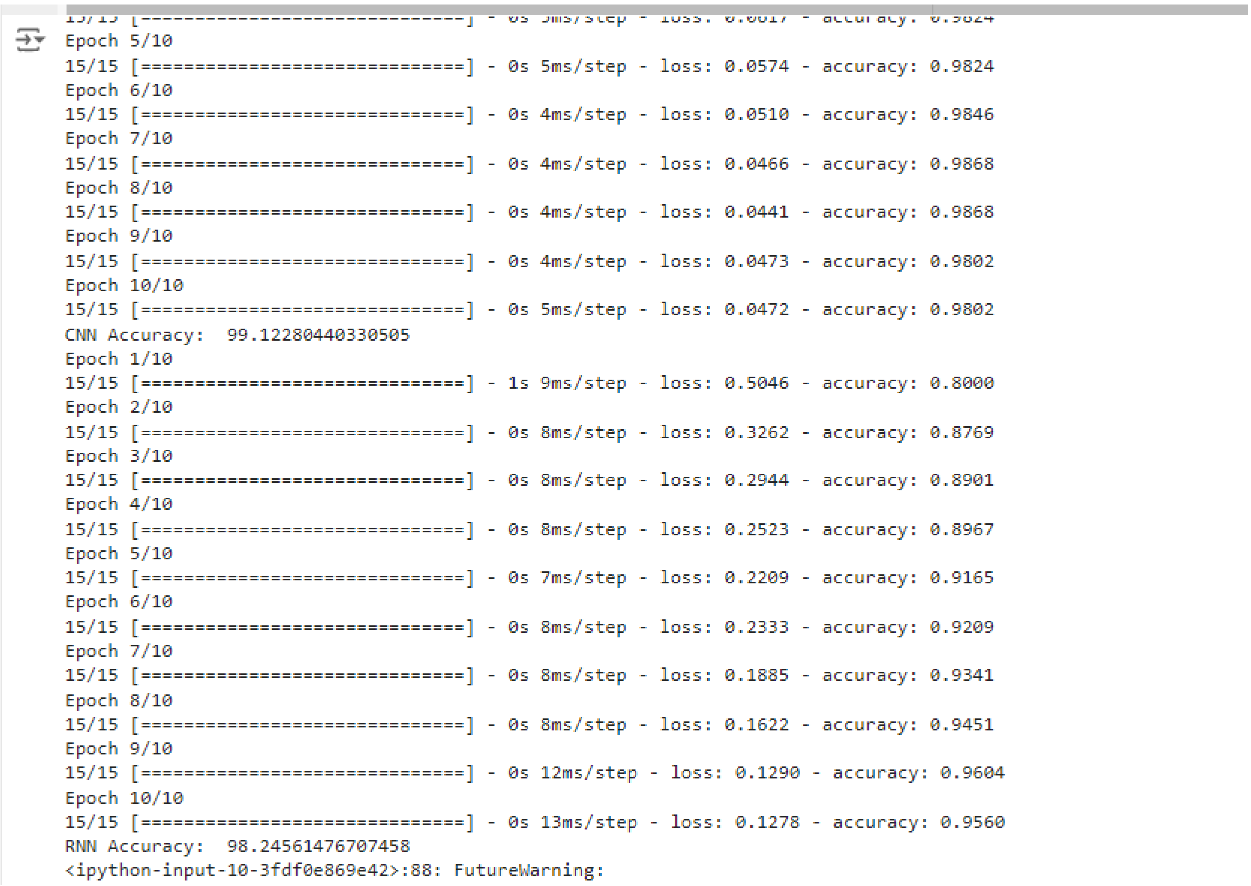


```
# Plot accuracies
models = ['Random Forest', 'KNN', 'SVM', 'CNN', 'RNN']
accuracies = [rf_acc * 100, knn_acc * 100, svm_acc * 100, cnn_acc * 100, rnn_acc * 100]

plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=accuracies, palette='Blues')
plt.ylabel('Accuracy %')
plt.title('Accuracy of Different Models')
plt.show()
```

OUTPUT:

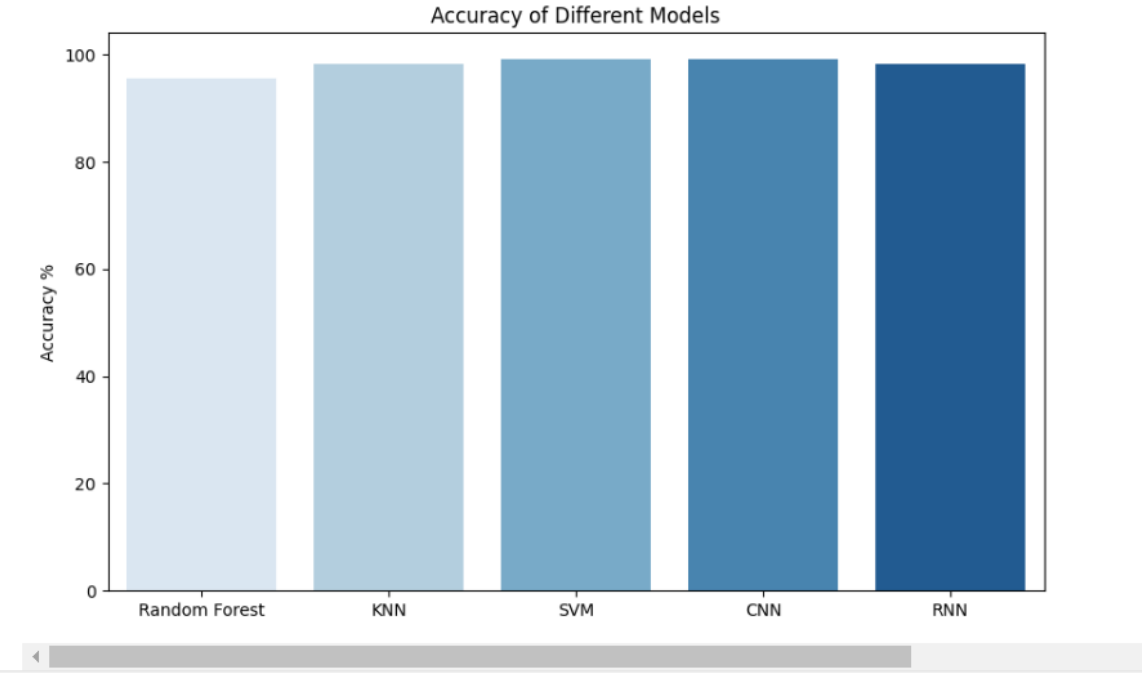


```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A

  sns.countplot(y, label='Count', palette='Blues')
Benign Tumors:  357
Malignant Tumors:  212
Random Forest Accuracy:  95.6140350877193
KNN Accuracy:  98.24561403508771
SVM Accuracy:  99.12280701754386
Epoch 1/10
15/15 [==============================] - 1s 4ms/step - loss: 0.3130 - accuracy: 0.8549
Epoch 2/10
15/15 [==============================] - 0s 5ms/step - loss: 0.1059 - accuracy: 0.9692
Epoch 3/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0747 - accuracy: 0.9802
Epoch 4/10
15/15 [==============================] - 0s 5ms/step - loss: 0.0617 - accuracy: 0.9824
Epoch 5/10
15/15 [==============================] - 0s 5ms/step - loss: 0.0574 - accuracy: 0.9824
Epoch 6/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0510 - accuracy: 0.9846
Epoch 7/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0466 - accuracy: 0.9868
Epoch 8/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0441 - accuracy: 0.9868
```



```
15/15 [==============================] - 0s 5ms/step - loss: 0.0617 - accuracy: 0.9824
Epoch 5/10
15/15 [==============================] - 0s 5ms/step - loss: 0.0574 - accuracy: 0.9824
Epoch 6/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0510 - accuracy: 0.9846
Epoch 7/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0466 - accuracy: 0.9868
Epoch 8/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0441 - accuracy: 0.9868
Epoch 9/10
15/15 [==============================] - 0s 4ms/step - loss: 0.0473 - accuracy: 0.9802
Epoch 10/10
15/15 [==============================] - 0s 5ms/step - loss: 0.0472 - accuracy: 0.9802
CNN Accuracy:  99.12280440330505
Epoch 1/10
15/15 [==============================] - 1s 9ms/step - loss: 0.5046 - accuracy: 0.8000
Epoch 2/10
15/15 [==============================] - 0s 8ms/step - loss: 0.3262 - accuracy: 0.8769
Epoch 3/10
15/15 [==============================] - 0s 8ms/step - loss: 0.2944 - accuracy: 0.8901
Epoch 4/10
15/15 [==============================] - 0s 8ms/step - loss: 0.2523 - accuracy: 0.8967
Epoch 5/10
15/15 [==============================] - 0s 7ms/step - loss: 0.2209 - accuracy: 0.9165
Epoch 6/10
15/15 [==============================] - 0s 8ms/step - loss: 0.2333 - accuracy: 0.9209
Epoch 7/10
15/15 [==============================] - 0s 8ms/step - loss: 0.1885 - accuracy: 0.9341
Epoch 8/10
15/15 [==============================] - 0s 8ms/step - loss: 0.1622 - accuracy: 0.9451
Epoch 9/10
15/15 [==============================] - 0s 12ms/step - loss: 0.1290 - accuracy: 0.9604
Epoch 10/10
15/15 [==============================] - 0s 13ms/step - loss: 0.1278 - accuracy: 0.9560
RNN Accuracy:  98.24561476707458
<ipython-input-10-3fdf0e869e42>:88: FutureWarning:
```

The bar graph plots the accuracy rates of the above mentioned 5 machine learning and deep learning algorithms.



Result:

The accuracies of each model are as follows:

-Random Forest: 95.61%

-KNN: 98.24%

-SVM: 99.12%

-CNN: 99.12%

-RNN: 98.24%

CONCLUSION

My internship at Jio Platform has been an immensely rewarding experience, allowing me to delve into the application of AI, ML, and DL algorithms for cancer detection. Throughout the one-month period, I had the opportunity to explore and implement a variety of advanced techniques, including Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). Each of these algorithms demonstrated unique strengths and challenges, providing me with a comprehensive understanding of their potential and limitations in the context of medical diagnostics.

By working on real-world data, I gained practical insights into the preprocessing, normalization, and evaluation processes essential for developing robust AI models. The hands-on experience of training, fine-tuning, and validating these models significantly enhanced my technical skills and deepened my appreciation for the complexities involved in deploying AI in healthcare.

This project underscored the critical role of AI and ML in improving early cancer detection and diagnosis. It also highlighted the importance of addressing data quality, model interpretability, and integration into clinical workflows to ensure the practical applicability of these technologies.

In conclusion, this internship has equipped me with valuable knowledge and experience that will be instrumental in my future endeavours. I am grateful for the opportunity to contribute to this impactful field and am motivated to continue exploring innovative solutions that leverage AI for the betterment of healthcare.