Topic	**Naive bayes**
Class Description	**Students learn the concepts of Naive bayes. They also learn about Bayes theorem. Students compare the Naive Bayes algorithm with Logistics regression and make conclusions.**
Class	**C120**
Class time	**45 mins**
Goal	● Explore the concept of Naive bayes algorithm. ● Create a prediction model using Naive bayes algorithm.
Resources Required	● Teacher Resources ○ Google Colab Notebook ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ● Student Resources ○ Google Colab Notebook ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen

Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

<u>CONTEXT</u>

- **Explore the concept of Naive Bayes algorithm and learn about bayes law.**

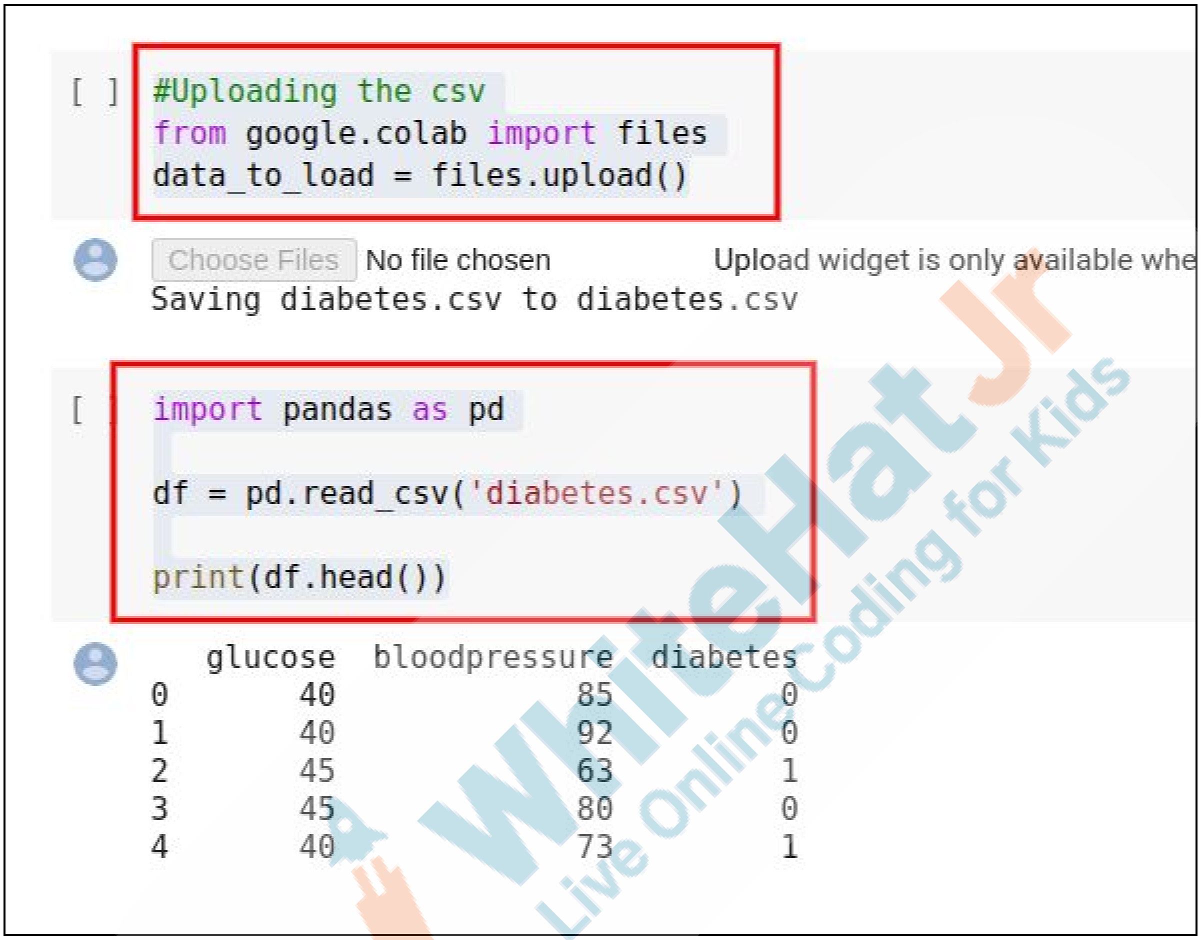
Class Steps	**Teacher Action**	**Student Action**
Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>! How are you doing today?	**ESR:** - We learned about the concept of Decision Tree. - We wrote a supervised

	Let's quickly revise what we did in last class?	learning algorithm called Decision Tree. - We also drew a prediction flow chart of the data.
	Awesome. So in last class we saw a supervised learning algorithm which made predictions from the decision rules it learnt from prior data training. Today we are going to look at another algorithm which assumes that every feature in a dataset is independent and has its own contribution to the outcome. Do you remember any other algorithm which we have studied that predicts the dependency of the variables in the dataset?	**ESR:** Logistics regression.
	Yes, so as you see Naive bayes and Logistic regression seem the same so many times people get confused on which algorithm to use. Before that let's learn more about the Naive bayes algorithm.	
Teacher Initiates Screen Share		
<u>**CHALLENGE**</u> ● **Create Naive bayes and logistics regression prediction models.** ● **See how each model performs on the basis of variable dependencies.** ● **Make a conclusion on the basis of the outcome.**		
Step 2: Teacher-led Activity (15 min)	Naive Bayes algorithm is a supervised machine learning algorithm based on the Bayes Probability theorem. Naive Bayes	*Student listens and asks questions.*

	assumes that there is no correlation between the features in a dataset used to train the model. Despite the oversimplified assumptions, Naive Bayes works very well in many real world complex problems. They require a relatively small number of training data samples to perform classification efficiently, compared to other algorithms like Logistic Regression and Decision trees, that we studied earlier.	
	Naive Bayes algorithm works on Bayes theorem. Bayes theorem or Bayes' law or Bayes' rule describes the probability of an event, based on prior knowledge of conditions that might be related to the event. What this means is that Bayes theorem describes the probability of a feature, based on prior knowledge of situations related to that feature. For example, if the probability of someone having diabetes is related to his or her age, then by using the Bayes Theorem, the age can be used to more accurately predict the probability of having diabetes.	*Student listens and asks questions.*
	The word naive implies that every pair of features in the dataset is independent of each other. Naive Bayes works on the assumption that	

	the value of a particular feature is independent of any other feature. For example, how can you classify if a vegetable is a tomato? Yes, but with Naive Bayes, each of these three features (shape, size and color) contributes independently to the probability that the vegetable is a tomato. Also, it assumes that there is no possible correlation between the shape, size and color.	**ESR:** A vegetable may be classified as a tomato if it's round, about 4-5 cm in diameter, and red in color.
	Let's write some code to understand the differences between the two as we try to understand Naive Bayes a little more. *<Teacher downloads the data from Teacher activity 1 and Opens the Google Colab Notebooks from Teacher activity 2>*	-
	<Teacher uploads the data and reads it using pandas and prints it> Here we are using the data for the causes of diabetes. Code:- **#Uploading the csv** **from google.colab import files** **data_to_load = files.upload()** **#Code to read the file.** **import pandas as pd** **df = pd.read_csv('diabetes.csv')**	*The student helps the teacher with the code to upload the data file, read it and print it's content.*

	print(df.head())	



```
[ ] #Uploading the csv
    from google.colab import files
    data_to_load = files.upload()
```

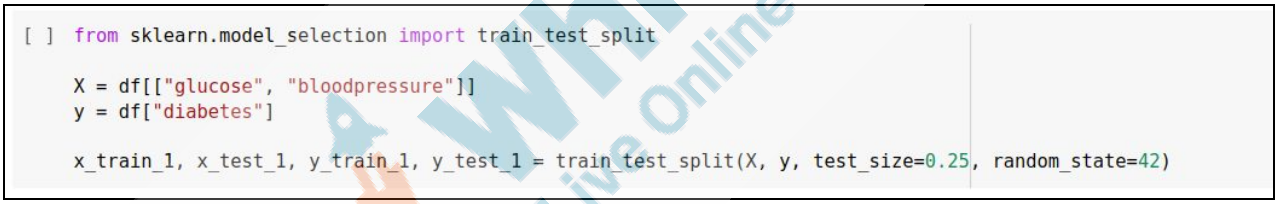
```
Saving diabetes.csv to diabetes.csv
```

```
[ ] import pandas as pd

    df = pd.read_csv('diabetes.csv')

    print(df.head())
```

```
   glucose  bloodpressure  diabetes
0       40             85         0
1       40             92         0
2       45             63         1
3       45             80         0
4       40             73         1
```

	In the data that we have, we can see that we have glucose, bloodpressure and we know if the given person has diabetes or not. Here, we will use the glucose and the bloodpressure to predict if the person has diabetes or not using Naive Bayes.	-

	Before that what is the first step we do with the data? Perfect!. *<Teacher codes to split the data for training and testing the model>* Code:- **from sklearn.model_selection import train_test_split** **X = df[["glucose", "bloodpressure"]]** **y = df["diabetes"]** **x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.25, random_state=42)**	**ESR:** We split the data into 2 parts to train and test the model. *The student helps the teacher to code for splitting the data to train and test the model.*

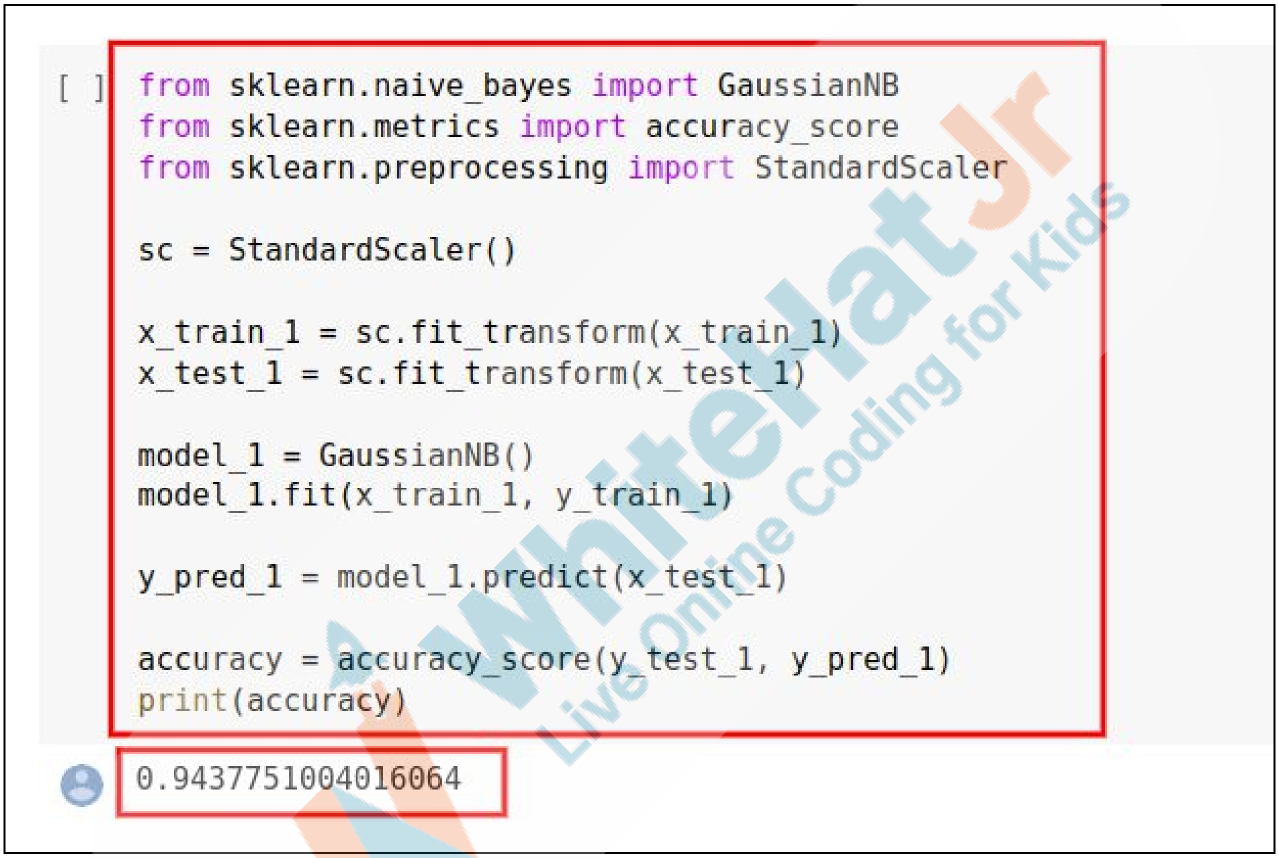


```
[ ]  from sklearn.model_selection import train_test_split

     X = df[["glucose", "bloodpressure"]]
     y = df["diabetes"]

     x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.25, random_state=42)
```

	Now we'll code to train the model with Naive Bayes. Code:- # first we are going to import GaussianNB module from sklearn naive_bayes module - Gaussian Naive Bayes algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e,	

	normal distribution. **from sklearn.naive_bayes import GaussianNB** **from sklearn.metrics import accuracy_score** **from sklearn.preprocessing import StandardScaler** **sc = StandardScaler()** **x_train_1 = sc.fit_transform(x_train_1)** **x_test_1 = sc.fit_transform(x_test_1)** **model_1 = GaussianNB()** **model_1.fit(x_train_1, y_train_1)** **y_pred_1 = model_1.predict(x_test_1)** **accuracy = accuracy_score(y_test_1, y_pred_1)** **print(accuracy)** Can you tell me what accuracy_score and StandardScaler do?	**ESR:** accuracy_score returns "accuracy classification score". What it does is the calculation of "How accurate the classification is". StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.

	Perfect. So what accuracy do we see here? Now let's see if we can get this accuracy using the logistics regression.	**ESR:** We can see an amazing accuracy of 94.4%.

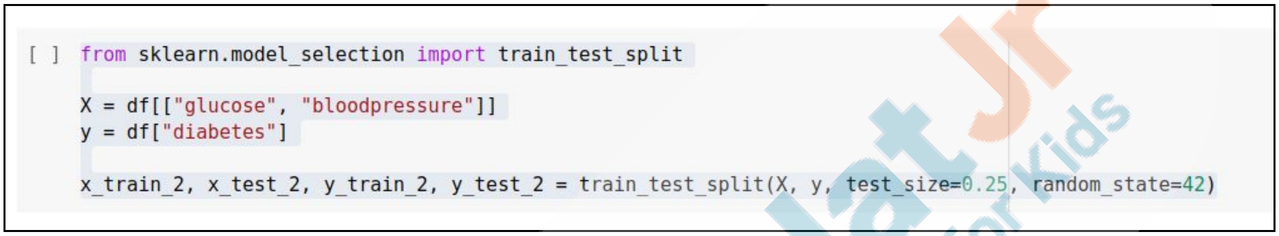


```
[ ] from sklearn.naive_bayes import GaussianNB
    from sklearn.metrics import accuracy_score
    from sklearn.preprocessing import StandardScaler

    sc = StandardScaler()

    x_train_1 = sc.fit_transform(x_train_1)
    x_test_1 = sc.fit_transform(x_test_1)

    model_1 = GaussianNB()
    model_1.fit(x_train_1, y_train_1)

    y_pred_1 = model_1.predict(x_test_1)

    accuracy = accuracy_score(y_test_1, y_pred_1)
    print(accuracy)
```

```
0.9437751004016064
```

	So let's split the data to train and test our logistics regression model. *<Teacher codes to split the data to train and test the model>* Code:- **from sklearn.model_selection import train_test_split**	*The student helps the teacher with the code.*

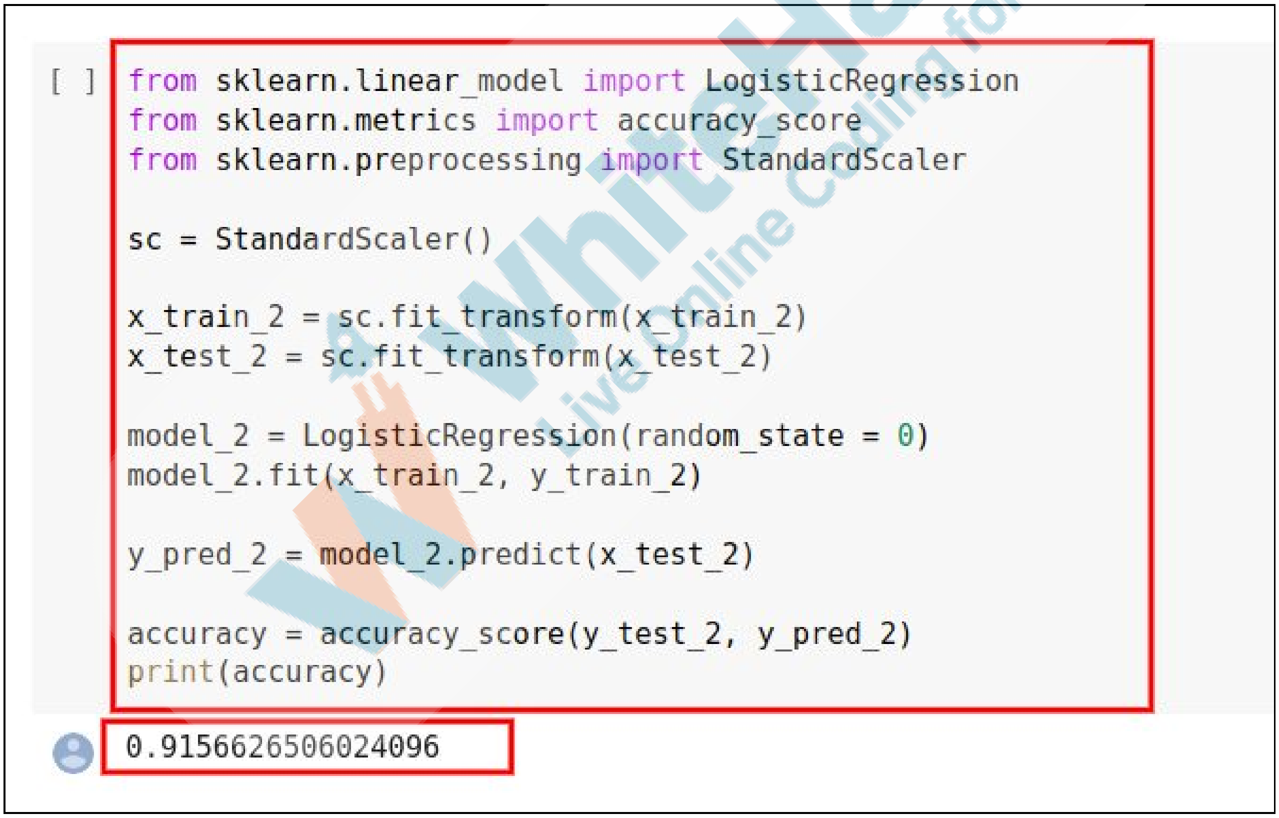
	X = df[["glucose", "bloodpressure"]] **y = df["diabetes"]** **x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(X, y, test_size=0.25, random_state=42)**	



```
[ ]  from sklearn.model_selection import train_test_split

     X = df[["glucose", "bloodpressure"]]
     y = df["diabetes"]

     x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(X, y, test_size=0.25, random_state=42)
```

	Now we have data ready, let's train our model on this data. *<Teacher codes to train the logistics regression model>* Code:- **from sklearn.linear_model import LogisticRegression** **from sklearn.metrics import accuracy_score** **from sklearn.preprocessing import StandardScaler** **sc = StandardScaler()** **x_train_2 = sc.fit_transform(x_train_2)** **x_test_2 = sc.fit_transform(x_test_2)** **model_2 =**	*The student helps the teacher with the code.*

	LogisticRegression(random_state = 0)
	model_2.fit(x_train_2, y_train_2)
	
	y_pred_2 = model_2.predict(x_test_2)
	
	accuracy = accuracy_score(y_test_2, y_pred_2)
	print(accuracy)
	
	What accuracy do you see?
		ESR:
		I can see an accuracy of 91.6%.



```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
    from sklearn.preprocessing import StandardScaler

    sc = StandardScaler()

    x_train_2 = sc.fit_transform(x_train_2)
    x_test_2 = sc.fit_transform(x_test_2)

    model_2 = LogisticRegression(random_state = 0)
    model_2.fit(x_train_2, y_train_2)

    y_pred_2 = model_2.predict(x_test_2)

    accuracy = accuracy_score(y_test_2, y_pred_2)
    print(accuracy)
```

```
0.9156626506024096
```

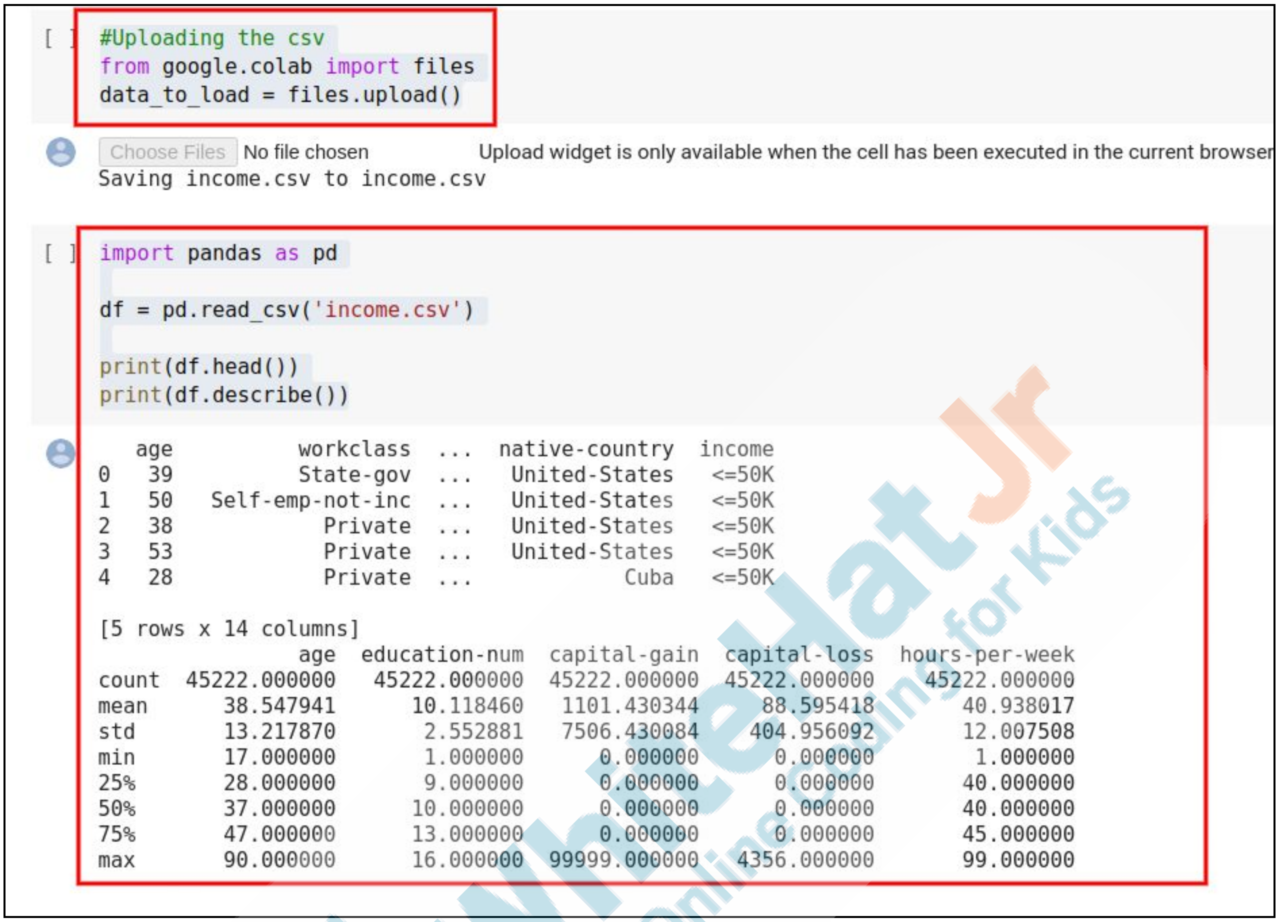
	While the accuracy score for both the datasets was close, with Naive Bayes giving us an accuracy of 94.4% and logistic regression giving us an	**ESR:** Varied

<table>
<tr><td></td><td>accuracy of 91.6%, Naive Bayes still performed better.

Can you guess why?

The reason for this is that if we look at our features again, we can see that the Glucose and the Blood Pressure had no correlation with each other. They both contribute individually to whether a person would have diabetes or not. This is exactly what Naive Bayes algorithm assumes, that all the features contribute individually to the outcome.</td><td></td></tr>
<tr><td></td><td>This was for the case of where Naive Bayes outperforms Logistic Regression, but let's see an example of the case where Logistic Regression outperforms Naive Bayes.
Can you try doing that? I'll guide you wherever you need help.</td><td><b>ESR:</b>
Yes!</td></tr>
<tr><td colspan="3"><b>Teacher Stops Screen Share</b></td></tr>
<tr><td></td><td>Now it's your turn. Please share your screen with me.</td><td></td></tr>
<tr><td colspan="3">• <b>Ask Student to press ESC key to come back to panel</b>
• <b>Guide Student to start Screen Share</b>
• <b>Teacher gets into Fullscreen</b></td></tr>
<tr><td colspan="3"><b><u>ACTIVITY</u></b>
• <b>Create Naive Bayes and logistics regression prediction models for different data.</b>
• <b>Make a conclusion on the basis of the outcome.</b></td></tr>
</table>

Step 3: Student-Led Activity (15 min)	*Teacher helps the student to download the data and open the Colab notebook.*	*Student downloads the data from Student Activity 1 and Opens Colab Notebook from Student Activity 2.*
	Here we are using the income data of various people. You can use pd.describe() to view some basic statistics details such as age, workclass etc. *<Teacher helps the student with the code>* Code:- **#Uploading the csv** **from google.colab import files** **data_to_load = files.upload()** **import pandas as pd** **df = pd.read_csv('income.csv')** **print(df.head())** **print(df.describe())**	*Student uploads the data and prints the data.*



```
[ ] #Uploading the csv
    from google.colab import files
    data_to_load = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser

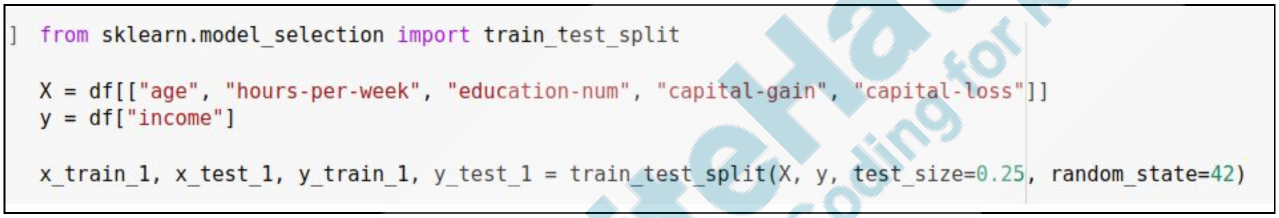
```
Saving income.csv to income.csv
```

```
[ ] import pandas as pd

    df = pd.read_csv('income.csv')

    print(df.head())
    print(df.describe())
```

```
   age         workclass  ...  native-country  income
0   39         State-gov  ...   United-States   <=50K
1   50  Self-emp-not-inc  ...   United-States   <=50K
2   38           Private  ...   United-States   <=50K
3   53           Private  ...   United-States   <=50K
4   28           Private  ...            Cuba   <=50K

[5 rows x 14 columns]
                age  education-num  capital-gain  capital-loss  hours-per-week
count  45222.000000   45222.000000  45222.000000  45222.000000    45222.000000
mean      38.547941      10.118460   1101.430344     88.595418       40.938017
std       13.217870       2.552881   7506.430084    404.956092       12.007508
min       17.000000       1.000000      0.000000      0.000000        1.000000
25%       28.000000       9.000000      0.000000      0.000000       40.000000
50%       37.000000      10.000000      0.000000      0.000000       40.000000
75%       47.000000      13.000000      0.000000      0.000000       45.000000
max       90.000000      16.000000  99999.000000   4356.000000       99.000000
```

	From the given data, we will consider the following fields to determine the salary of a person: Age Hours Per Week Education Number Capital Gain Capital Loss Now let's split the data to train and test the model. *<Teacher helps student to write code to split the model>*	*Student codes to split the data to train and test the model.*

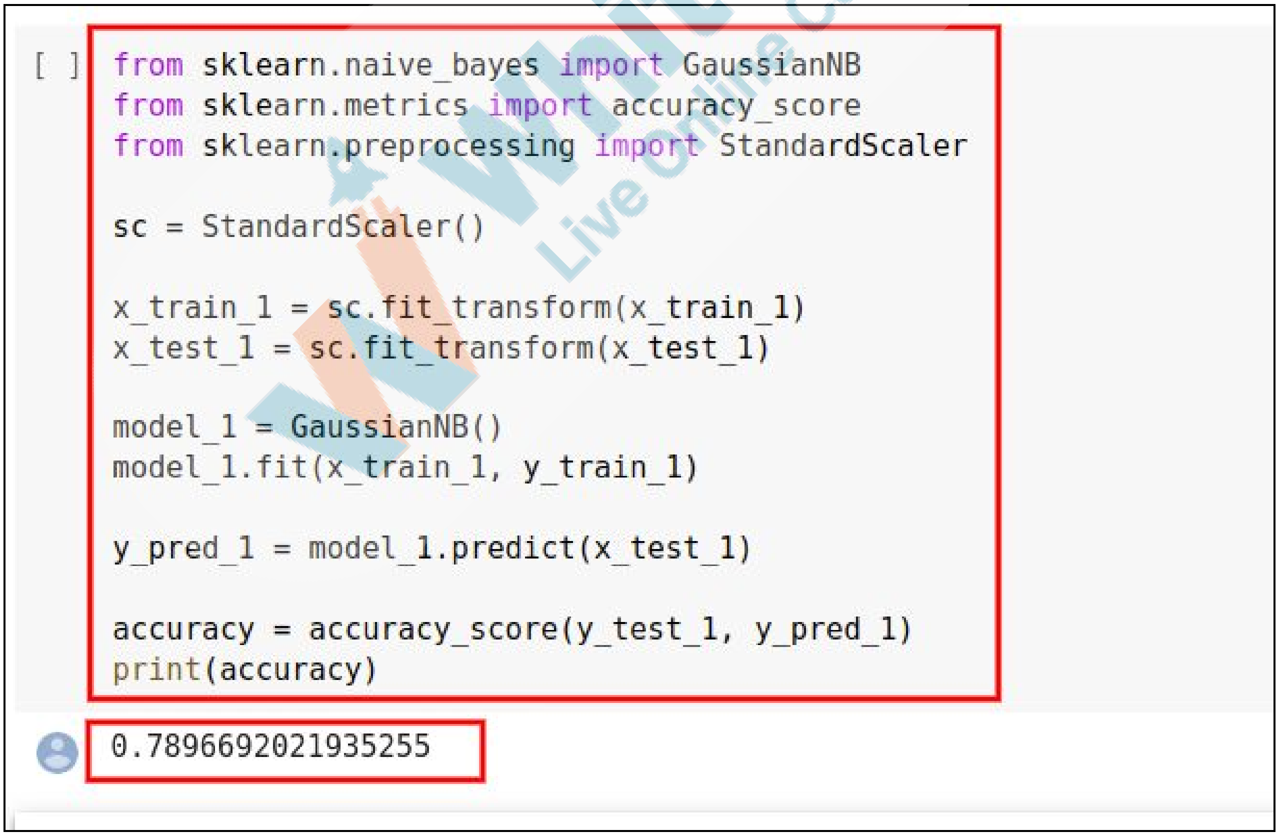
	Code:- **from sklearn.model_selection import train_test_split** **X = df[["age", "hours-per-week", "education-num", "capital-gain", "capital-loss"]]** **y = df["income"]** **x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.25, random_state=42)**	



```
from sklearn.model_selection import train_test_split

X = df[["age", "hours-per-week", "education-num", "capital-gain", "capital-loss"]]
y = df["income"]

x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(X, y, test_size=0.25, random_state=42)
```

	Now let's train the Naive Bayes model. *<Teacher helps student to code for training the Naive Bayes model>* Code:- **from sklearn.naive_bayes import GaussianNB** **from sklearn.metrics import accuracy_score** **from sklearn.preprocessing import StandardScaler** **sc = StandardScaler()** **x_train_1 =**	*The student codes to train the Naive Bayes model.*

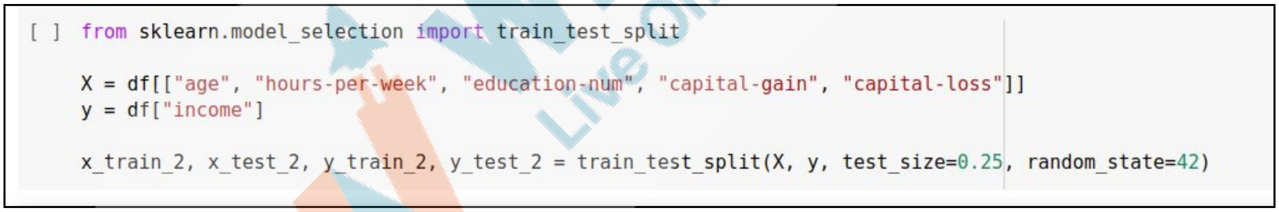
	sc.fit_transform(x_train_1) **x_test_1 =** **sc.fit_transform(x_test_1)** **model_1 = GaussianNB()** **model_1.fit(x_train_1, y_train_1)** **y_pred_1 =** **model_1.predict(x_test_1)** **accuracy =** **accuracy_score(y_test_1,** **y_pred_1)** **print(accuracy)** What accuracy can you see?	**ESR:** We can see the accuracy of almost 79%.



```
[ ]  from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import accuracy_score
     from sklearn.preprocessing import StandardScaler

     sc = StandardScaler()

     x_train_1 = sc.fit_transform(x_train_1)
     x_test_1 = sc.fit_transform(x_test_1)

     model_1 = GaussianNB()
     model_1.fit(x_train_1, y_train_1)

     y_pred_1 = model_1.predict(x_test_1)

     accuracy = accuracy_score(y_test_1, y_pred_1)
     print(accuracy)
```

```
0.7896692021935255
```

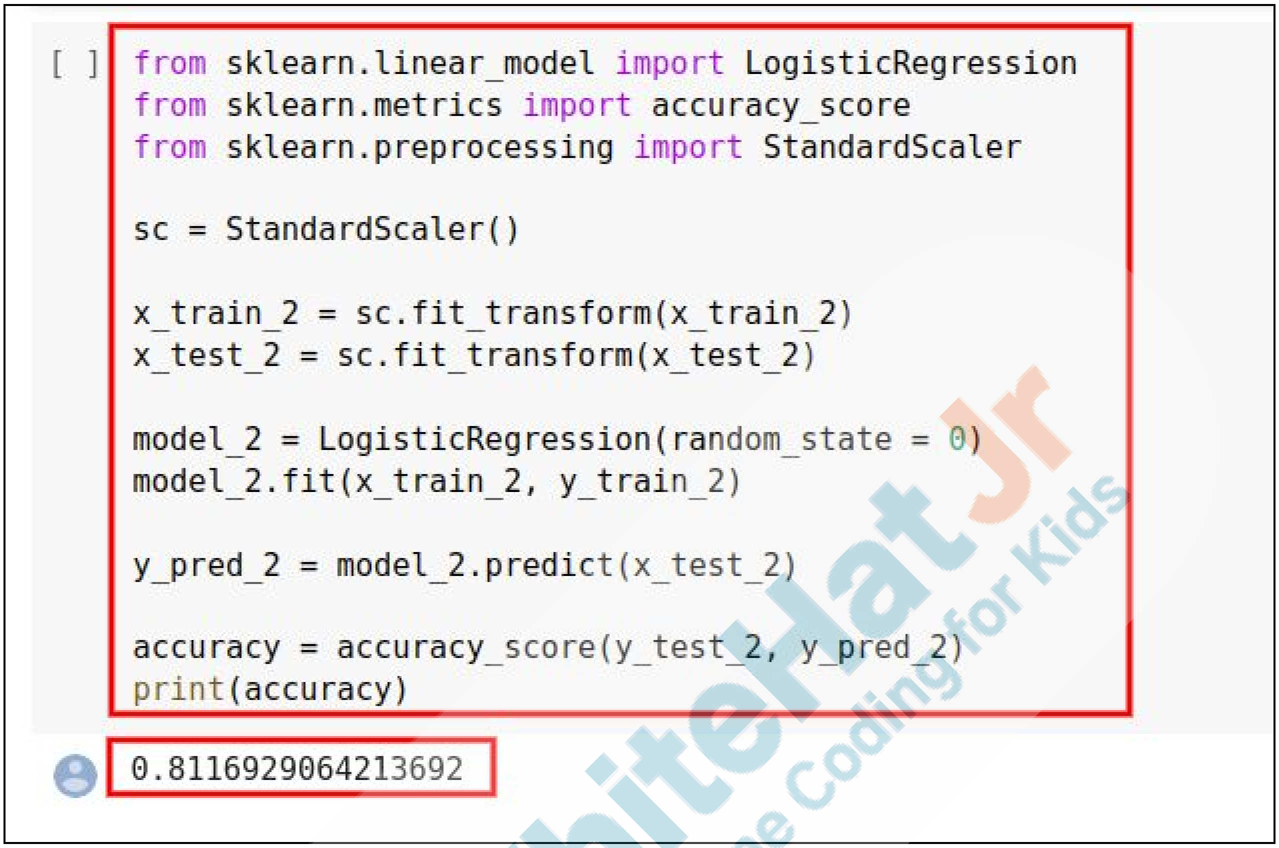
	Alright now let's check the accuracy with logistics regression. So let's split the data to train and test the logistics regression. *<Teacher helps the student to split the data>* Code: **from sklearn.model_selection import train_test_split** **X = df[["age", "hours-per-week", "education-num", "capital-gain", "capital-loss"]]** **y = df["income"]** **x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(X, y, test_size=0.25, random_state=42)**	*Student codes to split the data to train and test the model.*



```
[ ]  from sklearn.model_selection import train_test_split

     X = df[["age", "hours-per-week", "education-num", "capital-gain", "capital-loss"]]
     y = df["income"]

     x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(X, y, test_size=0.25, random_state=42)
```

	Now let's train the logistics regression model. *<Teacher helps student to code to train the model and print the accuracy>* Code: **from sklearn.linear_model import**	*<Student codes to train the model and print the accuracy>*

	LogisticRegression from sklearn.metrics import accuracy_score from sklearn.preprocessing import StandardScaler sc = StandardScaler() x_train_2 = sc.fit_transform(x_train_2) x_test_2 = sc.fit_transform(x_test_2) model_2 = LogisticRegression(random_state = 0) model_2.fit(x_train_2, y_train_2) y_pred_2 = model_2.predict(x_test_2) accuracy = accuracy_score(y_test_2, y_pred_2) print(accuracy)	



```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
    from sklearn.preprocessing import StandardScaler

    sc = StandardScaler()

    x_train_2 = sc.fit_transform(x_train_2)
    x_test_2 = sc.fit_transform(x_test_2)

    model_2 = LogisticRegression(random_state = 0)
    model_2.fit(x_train_2, y_train_2)

    y_pred_2 = model_2.predict(x_test_2)

    accuracy = accuracy_score(y_test_2, y_pred_2)
    print(accuracy)
```

```
0.8116929064213692
```

	Now what accuracy do we see? We can see that the logistics regression outperforms the Naive Bayes module.	**ESR:** We see the accuracy of 81.1%.
	In the first dataset, as we pointed out earlier, both the glucose and the blood pressure had little correlation, and both of them were contributing individually to whether a person has diabetes or not. Conclusion: In these kinds of dataset, where all the features contribute	

<table>
<tr><td></td><td>individually to the outcome, Naive Bayes outperforms logistic regression and is highly efficient.

In the second dataset, Logistic Regression outperformed Naive Bayes. The reason is that in this dataset, not all features contribute individually to the outcome. For example, there have been people of all age groups earning both less than and more than 50K. There have also been people with all education numbers that have an income of both less and more than 50K. Here, the combination of all the features is a better predictor of whether a person is earning more than or less than 50K, instead of all features having their individual contribution.</td><td></td></tr>
<tr><td colspan="3"><b>Teacher Guides Student to Stop Screen Share</b></td></tr>
<tr><td colspan="3"><b><u>FEEDBACK</u></b>
● <b>Appreciate the student for their efforts</b>
● <b>Identify 2 strengths and 1 area of progress for the student</b></td></tr>
<tr><td><b>Step 4:
Wrap-Up
(5 min)</b></td><td>So today we saw Naive Bayes algorithm for prediction.
Can you quickly revise what we did in today's class?</td><td><b>ESR:</b>
- We learned about the Naive bayes algorithm and theorem.
- We did a comparison between Naive bayes and logistics regression and saw how they outperform each</td></tr>
</table>

		other in different circumstances.
	The next class is going to be special! We will use our understanding of image processing techniques to create an invisibility cloak.	-
Project Overview	Naive Bayes **Goal of the Project:** In this project you will apply what you learned in the class and create your own algorithm. **Story:** Suppose you are working as a product manager at a wine factory, where you have to classify the product in various categories, you have large data of wines. Naive Bayes is the most straightforward and fast classification algorithm, which is suitable for a large chunk of data.Apply this algorithm to the data you have and note what interesting insights you find. I am very excited to see your project solution and I know you will do really well.	

	Bye Bye!	
	Teacher Clicks ✖ End Class	
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? - Describe what happened - Code I wrote ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflection in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Data for diabetes	https://raw.githubusercontent.com/whitehatjr/datasets/master/C120/diabetes.csv
Teacher Activity 2	Google Colab Notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true
Teacher Activity 3	Reference code	https://drive.google.com/file/d/1ZMw4bHrqhp69q-QtujVZVmHnA8xxlwuz/view?usp=sharing

Student Activity 1	Data for income	https://github.com/whitehatjr/datasets/blob/master/C120/income.csv
Student Activity 2	Google Colab Notebook	https://colab.research.google.com/notebooks/intro.ipynb#recent=true