كلية علوم الحاسب والمعلومات
College of Computer and Information Sciences

جامعة الإمام محمد بن سعود الإسلامية
IMAM MOHAMMAD IBN SAUD ISLAMIC UNIVERSITY

Report Project
Machine Learning (CS 360)

Project Overview:

In today's data-driven world, leveraging machine learning techniques has become imperative for solving complex problems and extracting valuable insights from data. This report presents the findings of our team's exploration into diagnosing patients and detecting diseases such as diabetes. These technological tools analyze vast amounts of medical data, including patient history, symptoms, and test results, to provide insights and support clinical decision-making. With machine learning algorithms and data-driven approaches, healthcare professionals can achieve more accurate diagnoses, personalized treatment plans, and improved patient outcomes. Additionally, wearable devices and remote monitoring systems allow for continuous health tracking, facilitating early detection and proactive management of conditions like diabetes. As technology continues to evolve, it holds the promise of further enhancing medical care and transforming the way we approach healthcare delivery
With the exponential growth of data availability, there is a pressing need to develop robust models capable of extracting meaningful patterns and making accurate predictions In this context, our team embarked on a journey to build and developing a model with the ability to predict whether a patient has diabetes or not based on
several factors by harnessing the power of machine learning algorithms

Dataset Description :

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases

The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset. Several constraints were placed
on the selection of these instances from a larger database. In particular all patients here are females at least 21 years old of Pima Indian heritage

Information about dataset features :

Pregnancies: To express the Number of pregnancies 0 to 17

Glucose: To express the Glucose level in blood 0 to 199

BloodPressure: To express the Blood pressure measurement 0 to 122

SkinThickness: To express the thickness of the skin 0 to 99

Insulin: To express the Insulin level in blood 0 to 846

BMI: To express the Body mass index 0 to 67.1

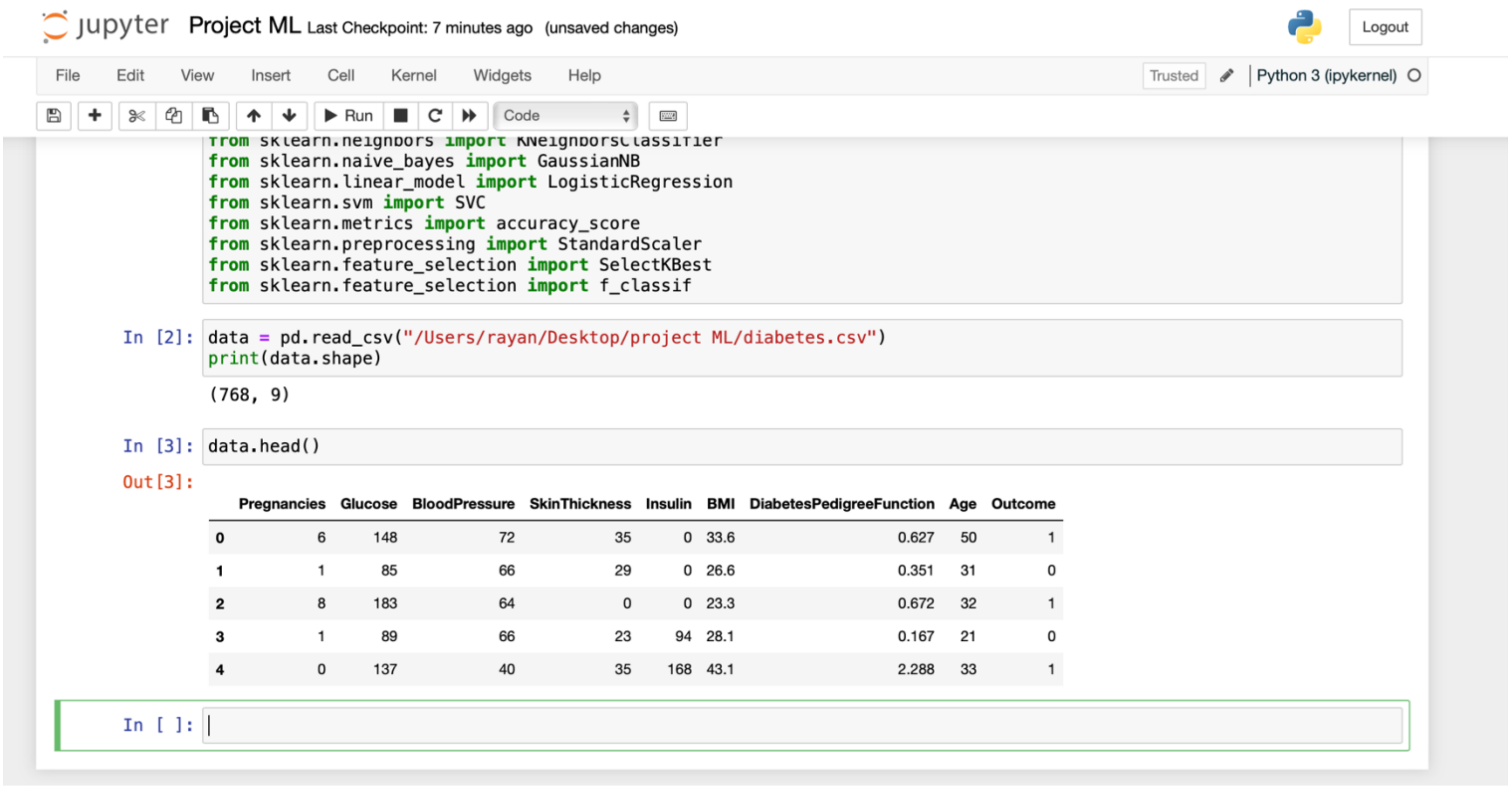
DiabetesPedigreeFunction: To express the Diabetes percentage 0.08 to 2.42

Age: To express the age 21 to 81

The target of the dataset that determine the result :

Outcome: To express the final result 1 is Yes and 0 is No

Visual figure of Features/Attributes and Target with five observations



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

```
In [2]: data = pd.read_csv("/Users/rayan/Desktop/project ML/diabetes.csv")
        print(data.shape)

        (768, 9)

In [3]: data.head()

Out[3]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [ ]:
```

Machine Learning Model Selection:

The selection of these algorithms was meticulously conducted, drawing from a deep understanding of the underlying data set earmarked for model training

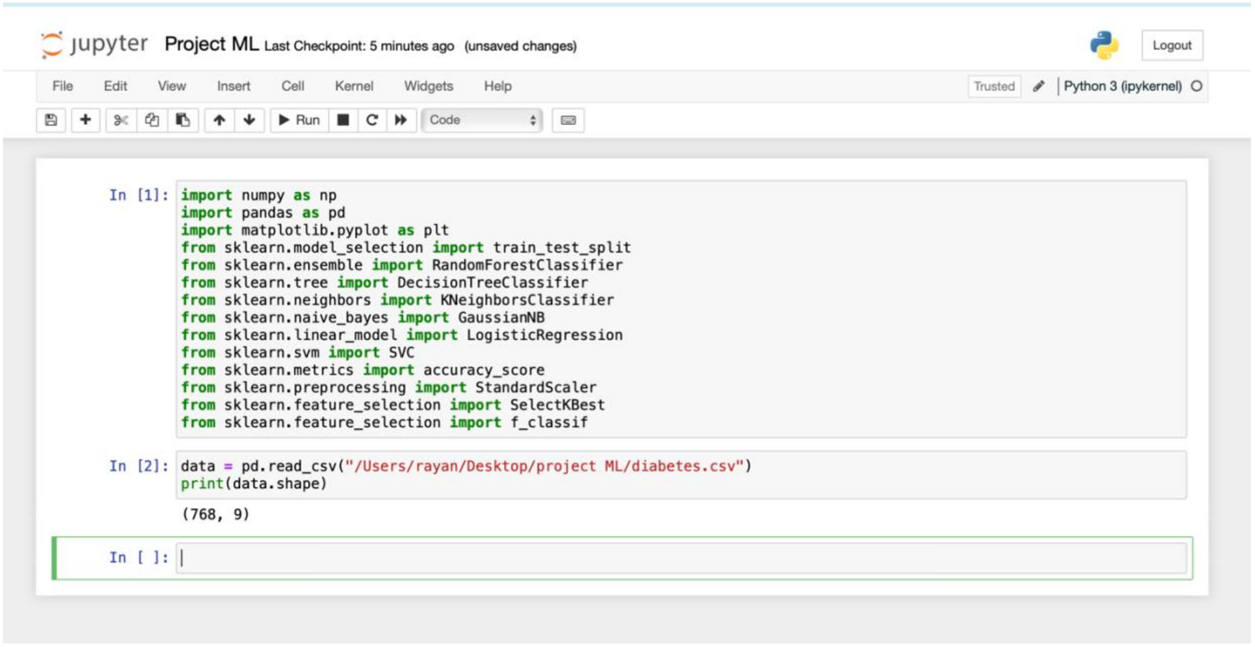
Random Forest and SVM, in particular, were highlighted for their robust performance in such scenarios, underpinned by their ability to handle complex data relationships and deliver accurate predictions. The inherent flexibility of Random Forest in capturing intricate patterns within the data, coupled with the discriminative power of SVM in separating binary classes with a clear margin, made them prime candidates for our analysis

However, recognizing the diverse nature of our data and the inherent variability in algorithm performance, we ensured a comprehensive approach by incorporating additional models like Logistic Regression, Naive Bayes, and kNN. By doing so, we aimed to harness the unique strengths of each algorithm, providing a holistic evaluation of their accuracy and efficacy in our specific context.

Implementation:

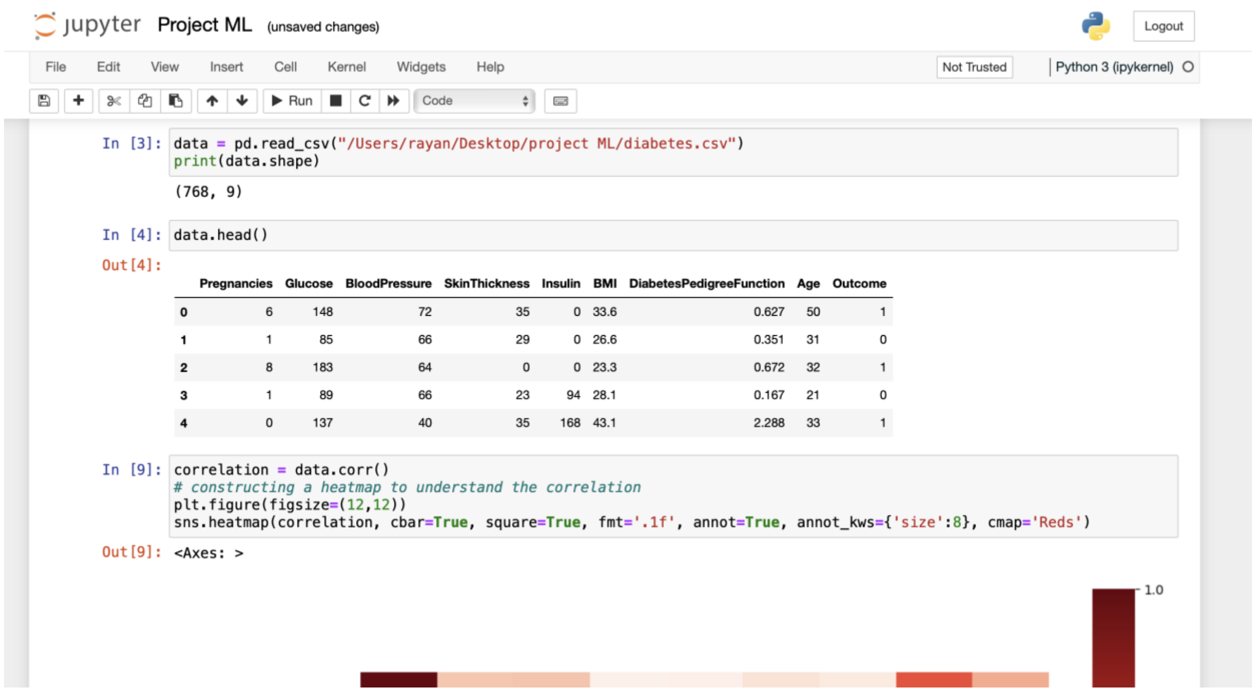
During this project, we used JupyterLab as our framework of choice. JupyterLab was instrumental in creating a conducive environment for our work, facilitating seamless integration with the required libraries essential for executing our tasks and training our models. Its versatility and user-friendly interface allowed us to import and leverage the necessary libraries effortlessly, streamlining the development and experimentation process. By harnessing the capabilities of JupyterLab, we were able to maintain an efficient workflow, ensuring smooth execution of our project objectives.

Initially, we will review the code along with an explanation of each part:


jupyter Project ML Last Checkpoint: 5 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.naive_bayes import GaussianNB
        from sklearn.linear_model import LogisticRegression
        from sklearn.svm import SVC
        from sklearn.metrics import accuracy_score
        from sklearn.preprocessing import StandardScaler
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import f_classif

In [2]: data = pd.read_csv("/Users/rayan/Desktop/project ML/diabetes.csv")
        print(data.shape)

        (768, 9)

In [ ]:
```

In the first part after we have imported required library we counted features and observations as you can see

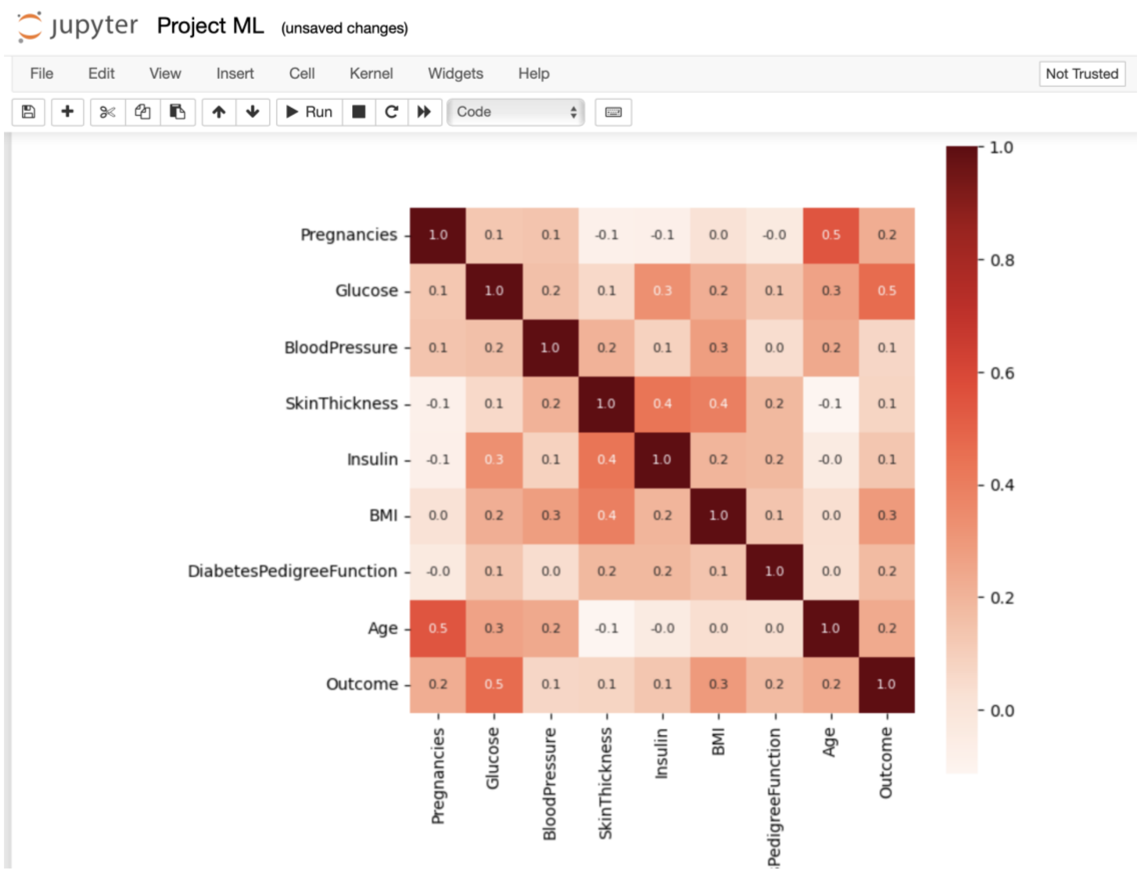

jupyter Project ML (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [3]: data = pd.read_csv("/Users/rayan/Desktop/project ML/diabetes.csv")
        print(data.shape)

        (768, 9)

In [4]: data.head()

Out[4]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

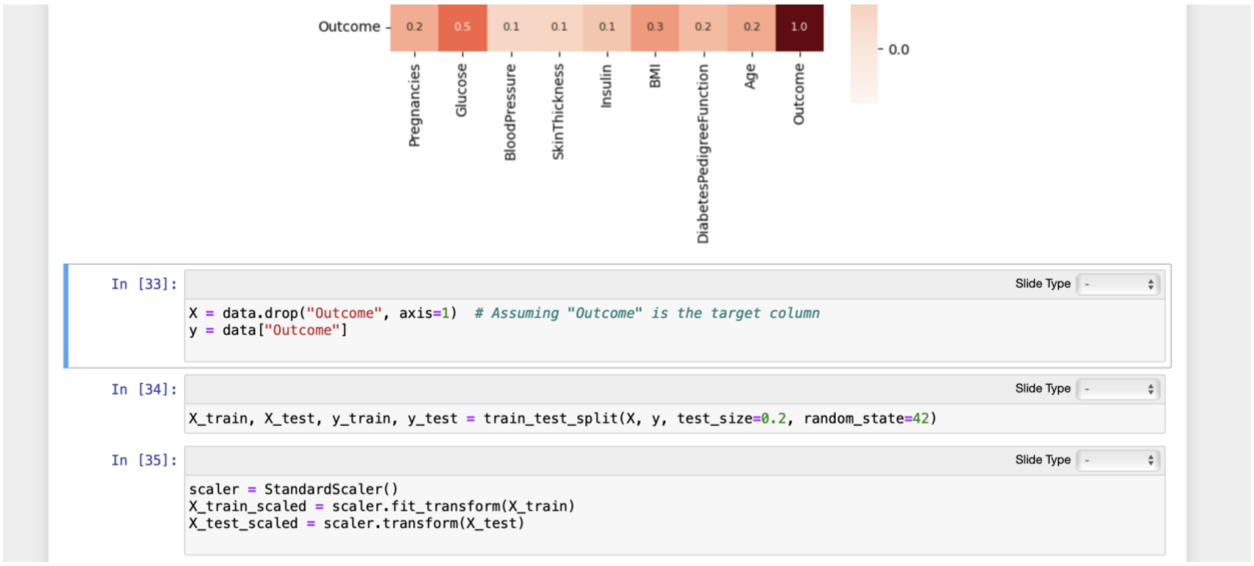
```
In [9]: correlation = data.corr()
        # constructing a heatmap to understand the correlation
        plt.figure(figsize=(12,12))
        sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Reds')

Out[9]: <Axes: >
```

Here, we have presented a visual representation of the data and we constructed heatmap for showing the correlation



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	PedigreeFunction	Age	Outcome
Pregnancies	1.0	0.1	0.1	-0.1	-0.1	0.0	-0.0	0.5	0.2
Glucose	0.1	1.0	0.2	0.1	0.3	0.2	0.1	0.3	0.5
BloodPressure	0.1	0.2	1.0	0.2	0.1	0.3	0.0	0.2	0.1
SkinThickness	-0.1	0.1	0.2	1.0	0.4	0.4	0.2	-0.1	0.1
Insulin	-0.1	0.3	0.1	0.4	1.0	0.2	0.2	-0.0	0.1
BMI	0.0	0.2	0.3	0.4	0.2	1.0	0.1	0.0	0.3
DiabetesPedigreeFunction	-0.0	0.1	0.0	0.2	0.2	0.1	1.0	0.0	0.2
Age	0.5	0.3	0.2	-0.1	-0.0	0.0	0.0	1.0	0.2
Outcome	0.2	0.5	0.1	0.1	0.1	0.3	0.2	0.2	1.0

And in next image we split data into two set X refer to features and y refer to target



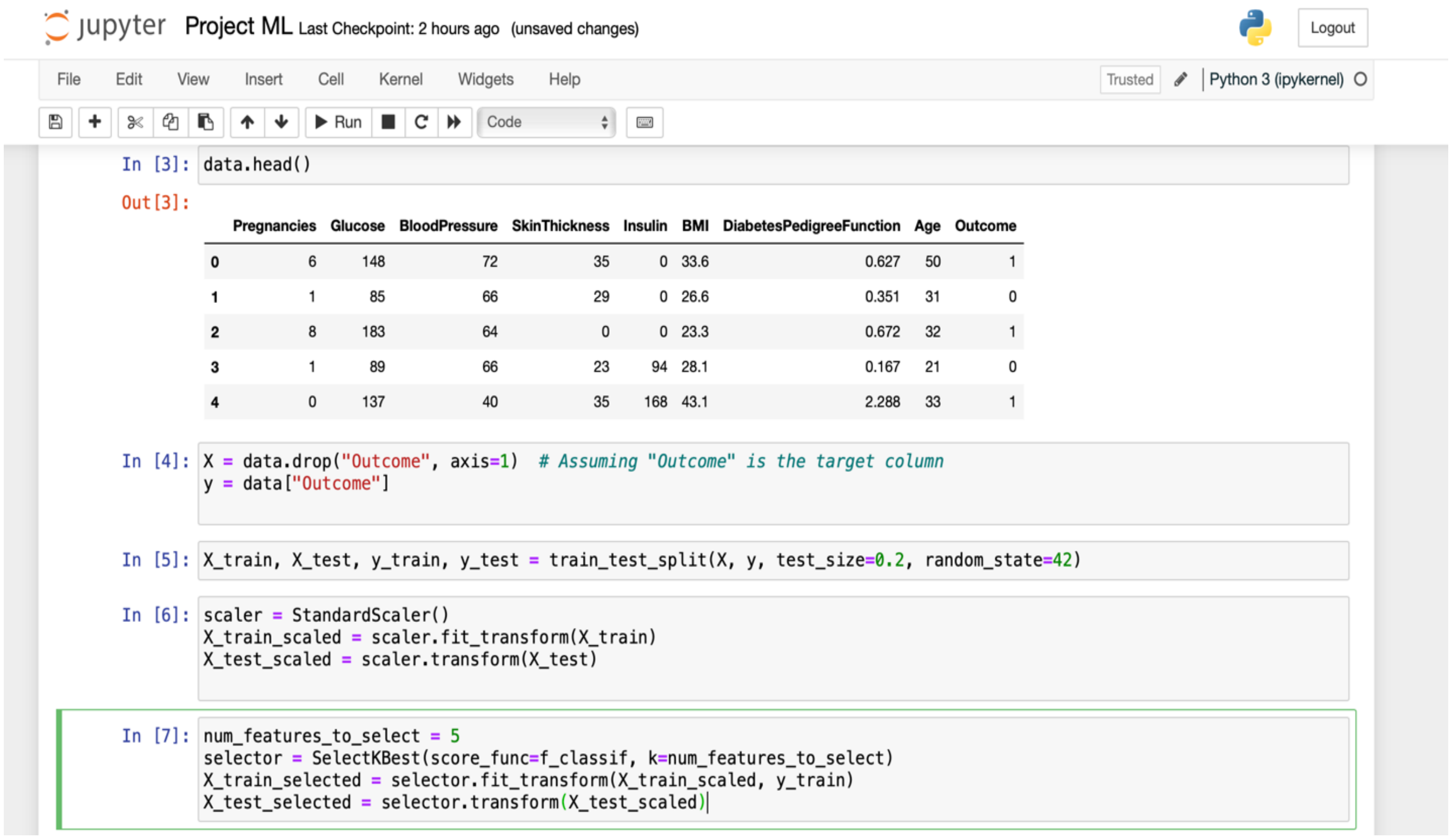
In [33]:
```
X = data.drop("Outcome", axis=1)  # Assuming "Outcome" is the target column
y = data["Outcome"]
```

In [34]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [35]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In previous , **at part 34**, we divided the data into two sets, as evident in the image: one for training and one for testing

And **at the part 35**, we applied feature scaling to streamline the data and enhance the ease of model training.



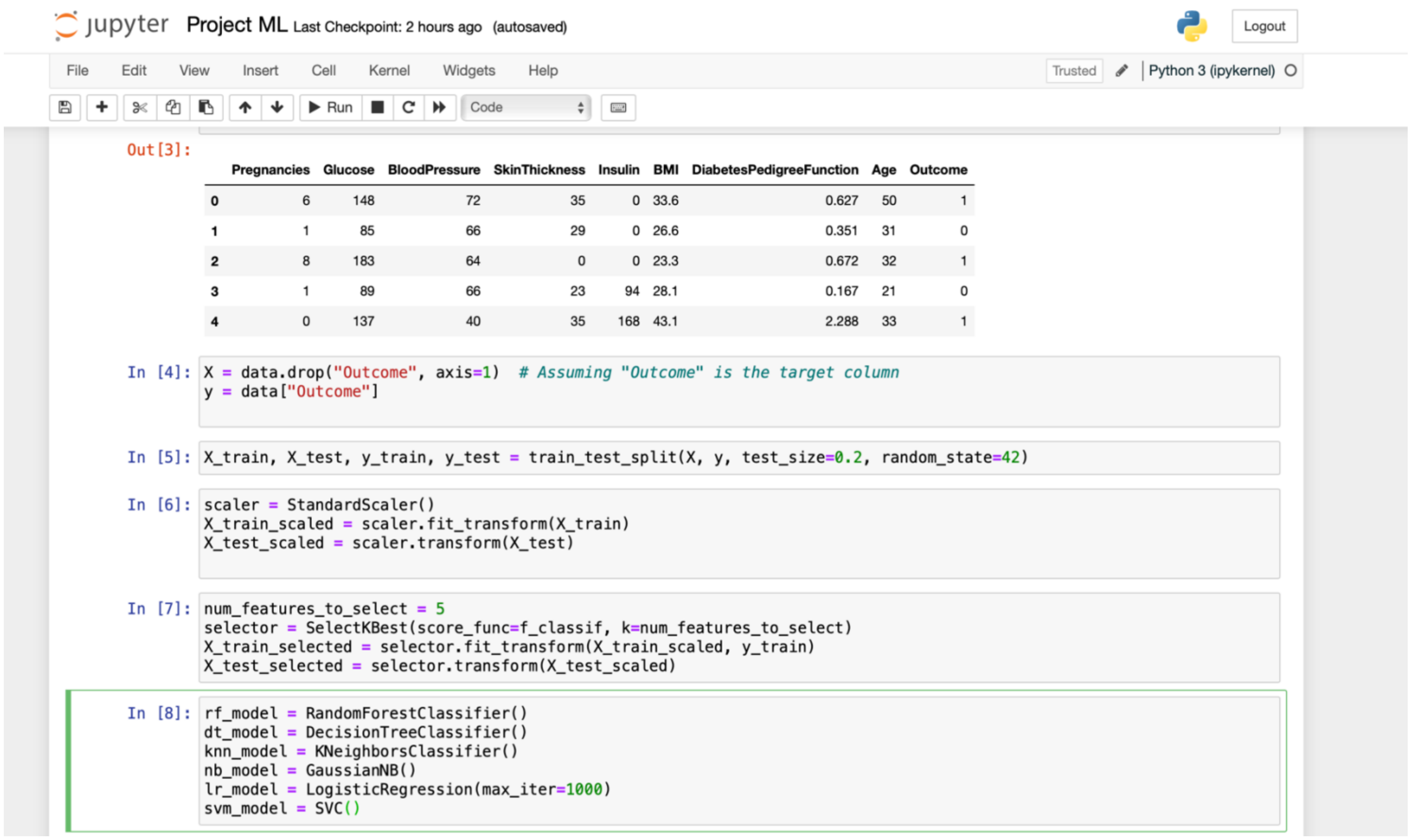
```
In [3]: data.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [4]: X = data.drop("Outcome", axis=1)  # Assuming "Outcome" is the target column
        y = data["Outcome"]

In [5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [6]: scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

In [7]: num_features_to_select = 5
        selector = SelectKBest(score_func=f_classif, k=num_features_to_select)
        X_train_selected = selector.fit_transform(X_train_scaled, y_train)
        X_test_selected = selector.transform(X_test_scaled)
```

At at part 7 , these perform feature selection based on correlation using the SelectKBest method with f_classif as the scoring function. It selects the top num_features_to_select features with the highest correlation with the target variable


jupyter Project ML Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```
X = data.drop("Outcome", axis=1)  # Assuming "Outcome" is the target column
y = data["Outcome"]
```

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

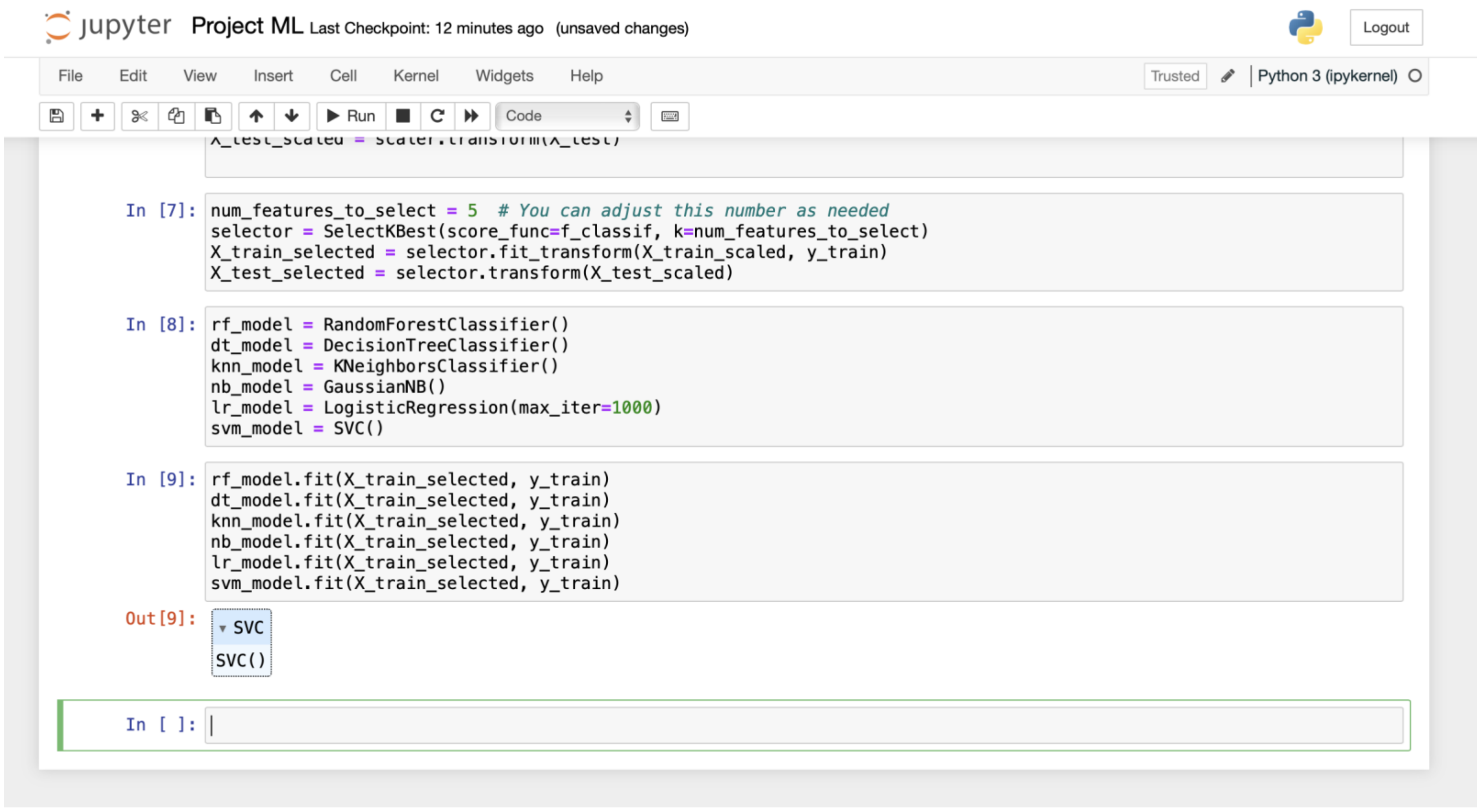
In [6]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [7]:
```
num_features_to_select = 5
selector = SelectKBest(score_func=f_classif, k=num_features_to_select)
X_train_selected = selector.fit_transform(X_train_scaled, y_train)
X_test_selected = selector.transform(X_test_scaled)
```

In [8]:
```
rf_model = RandomForestClassifier()
dt_model = DecisionTreeClassifier()
knn_model = KNeighborsClassifier()
nb_model = GaussianNB()
lr_model = LogisticRegression(max_iter=1000)
svm_model = SVC()
```

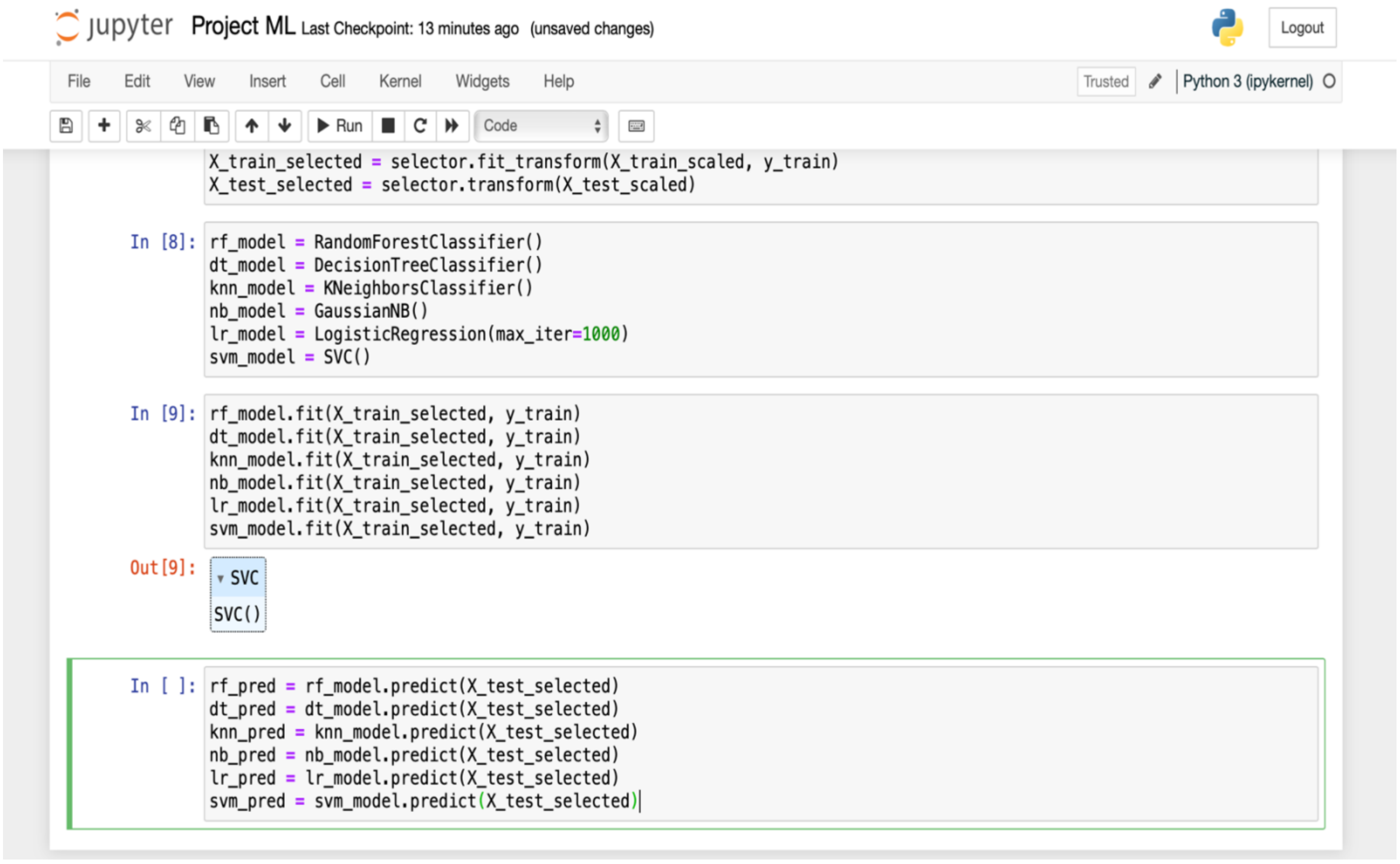
At the step 8 Initializing machine learning models for classification:

- rf_model: A Random Forest Classifier.
- dt_model: A Decision Tree Classifier.
- knn_model: A K-Nearest Neighbors Classifier.
- nb_model: A Gaussian Naive Bayes Classifier.
- lr_model: A Logistic Regression Classifier with an increased max_iter parameter set to 1000 to avoid convergence issues
- svm_model: A Support Vector Machine Classifier.



```
In [7]: num_features_to_select = 5  # You can adjust this number as needed
        selector = SelectKBest(score_func=f_classif, k=num_features_to_select)
        X_train_selected = selector.fit_transform(X_train_scaled, y_train)
        X_test_selected = selector.transform(X_test_scaled)

In [8]: rf_model = RandomForestClassifier()
        dt_model = DecisionTreeClassifier()
        knn_model = KNeighborsClassifier()
        nb_model = GaussianNB()
        lr_model = LogisticRegression(max_iter=1000)
        svm_model = SVC()

In [9]: rf_model.fit(X_train_selected, y_train)
        dt_model.fit(X_train_selected, y_train)
        knn_model.fit(X_train_selected, y_train)
        nb_model.fit(X_train_selected, y_train)
        lr_model.fit(X_train_selected, y_train)
        svm_model.fit(X_train_selected, y_train)

Out[9]: ▾ SVC
        SVC()

In [ ]:
```

This image clearly illustrates the training of the models on


Jupyter Project ML Last Checkpoint: 13 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
X_train_selected = selector.fit_transform(X_train_scaled, y_train)
X_test_selected = selector.transform(X_test_scaled)
```

In [8]:
```
rf_model = RandomForestClassifier()
dt_model = DecisionTreeClassifier()
knn_model = KNeighborsClassifier()
nb_model = GaussianNB()
lr_model = LogisticRegression(max_iter=1000)
svm_model = SVC()
```

In [9]:
```
rf_model.fit(X_train_selected, y_train)
dt_model.fit(X_train_selected, y_train)
knn_model.fit(X_train_selected, y_train)
nb_model.fit(X_train_selected, y_train)
lr_model.fit(X_train_selected, y_train)
svm_model.fit(X_train_selected, y_train)
```

Out[9]:
```
▾ SVC
SVC()
```

In [ ]:
```
rf_pred = rf_model.predict(X_test_selected)
dt_pred = dt_model.predict(X_test_selected)
knn_pred = knn_model.predict(X_test_selected)
nb_pred = nb_model.predict(X_test_selected)
lr_pred = lr_model.predict(X_test_selected)
svm_pred = svm_model.predict(X_test_selected)
```

Here, as demonstrated in the final section, we proceeded to make predictions using the trained models.

Evaluation & Results Discussion :

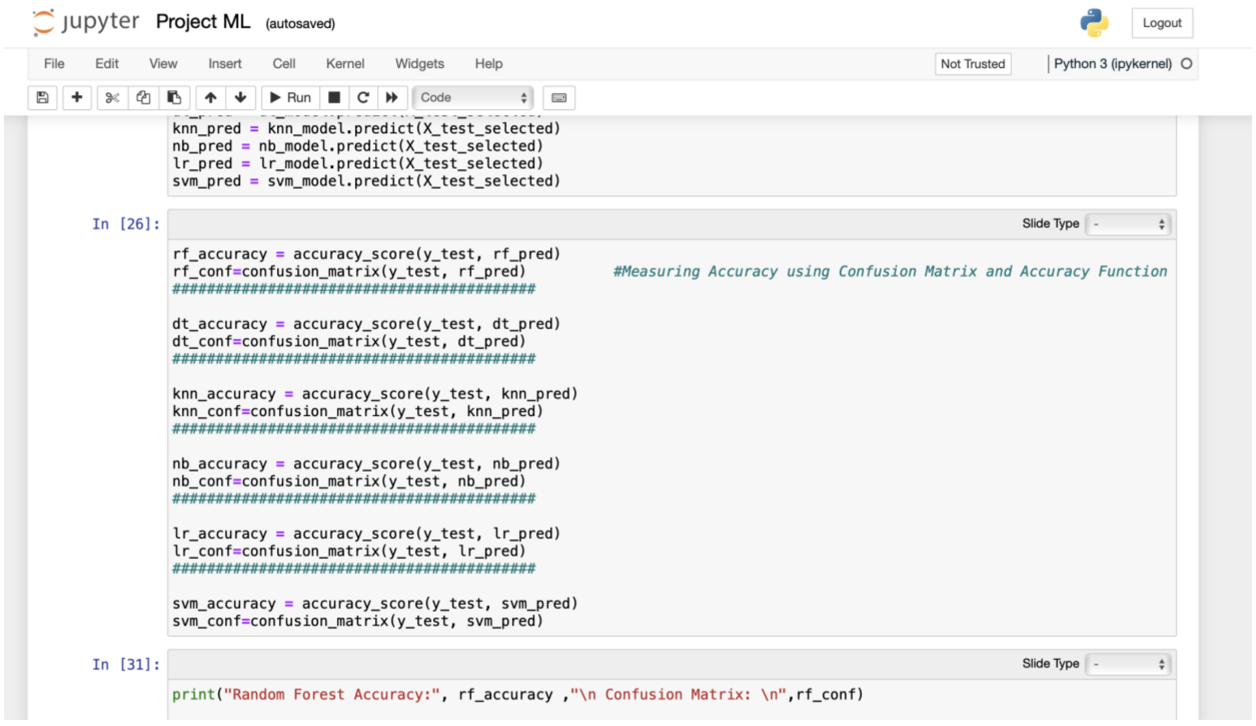
In order to comprehensively evaluate the efficacy of the models trained during the course of our project and to precisely measure their accuracy, we opted to create a detailed visual representation in the form of a bar chart. This strategic visualization not only facilitates a comparative analysis of the performance of each model but also provides a clear insight into which model yields the highest level of accuracy.

The process began with the construction of an array specifically dedicated to labeling, ensuring clarity and coherence in the representation of the different models under evaluation. Subsequently, another array was meticulously crafted to capture and encapsulate the accuracy metrics of each individual model. This meticulous approach allows for a granular examination of the performance of each model, facilitating nuanced insights into their respective strengths and weaknesses.

By leveraging this comprehensive framework, we are able to present a holistic overview of the performance of the trained models, thereby empowering stakeholders to make informed decisions regarding model selection and deployment. The resultant bar chart serves as a powerful visual aid, offering a

compelling snapshot of the comparative accuracy levels attained by each model

Firstly , let's see how to measure the accuracy of the trained models is , as indicated in the following image :



```
knn_pred = knn_model.predict(X_test_selected)
nb_pred = nb_model.predict(X_test_selected)
lr_pred = lr_model.predict(X_test_selected)
svm_pred = svm_model.predict(X_test_selected)
```

In [26]:

```
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_conf=confusion_matrix(y_test, rf_pred)          #Measuring Accuracy using Confusion Matrix and Accuracy Function
##########################################

dt_accuracy = accuracy_score(y_test, dt_pred)
dt_conf=confusion_matrix(y_test, dt_pred)
##########################################

knn_accuracy = accuracy_score(y_test, knn_pred)
knn_conf=confusion_matrix(y_test, knn_pred)
##########################################

nb_accuracy = accuracy_score(y_test, nb_pred)
nb_conf=confusion_matrix(y_test, nb_pred)
##########################################

lr_accuracy = accuracy_score(y_test, lr_pred)
lr_conf=confusion_matrix(y_test, lr_pred)
##########################################

svm_accuracy = accuracy_score(y_test, svm_pred)
svm_conf=confusion_matrix(y_test, svm_pred)
```

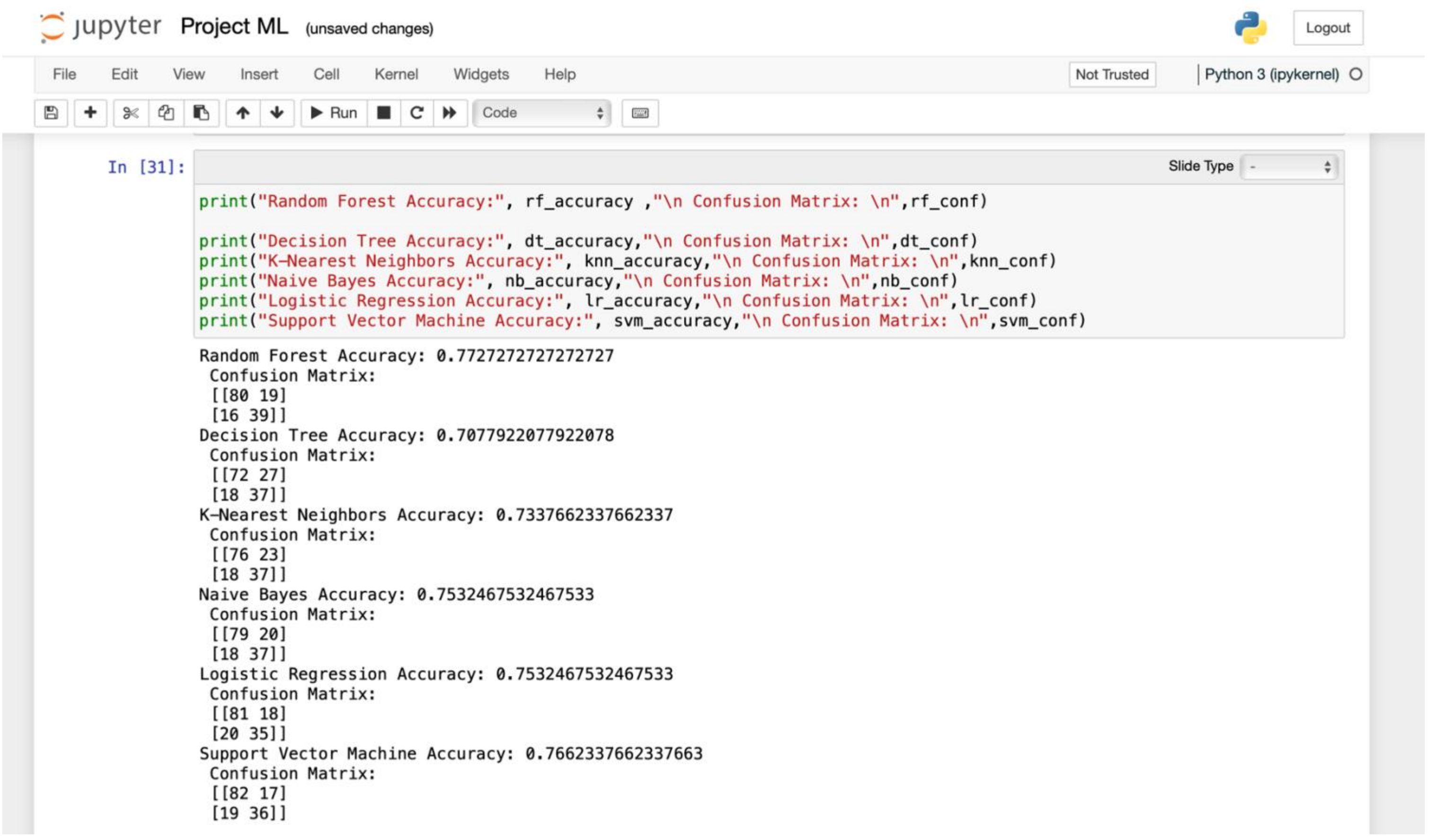
In [31]:

```
print("Random Forest Accuracy:", rf_accuracy ,"\n Confusion Matrix: \n",rf_conf)
```

In this part, we evaluated the models by first measuring the accuracy of each model using the accuracy_score function and Confusion Matrix we do not want to rely on one source by measuring the accuracy

at part 26, we invoked the functions and passed the outputs of the models and the original outputs to it

----


jupyter Project ML (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Slide Type -

In [31]:

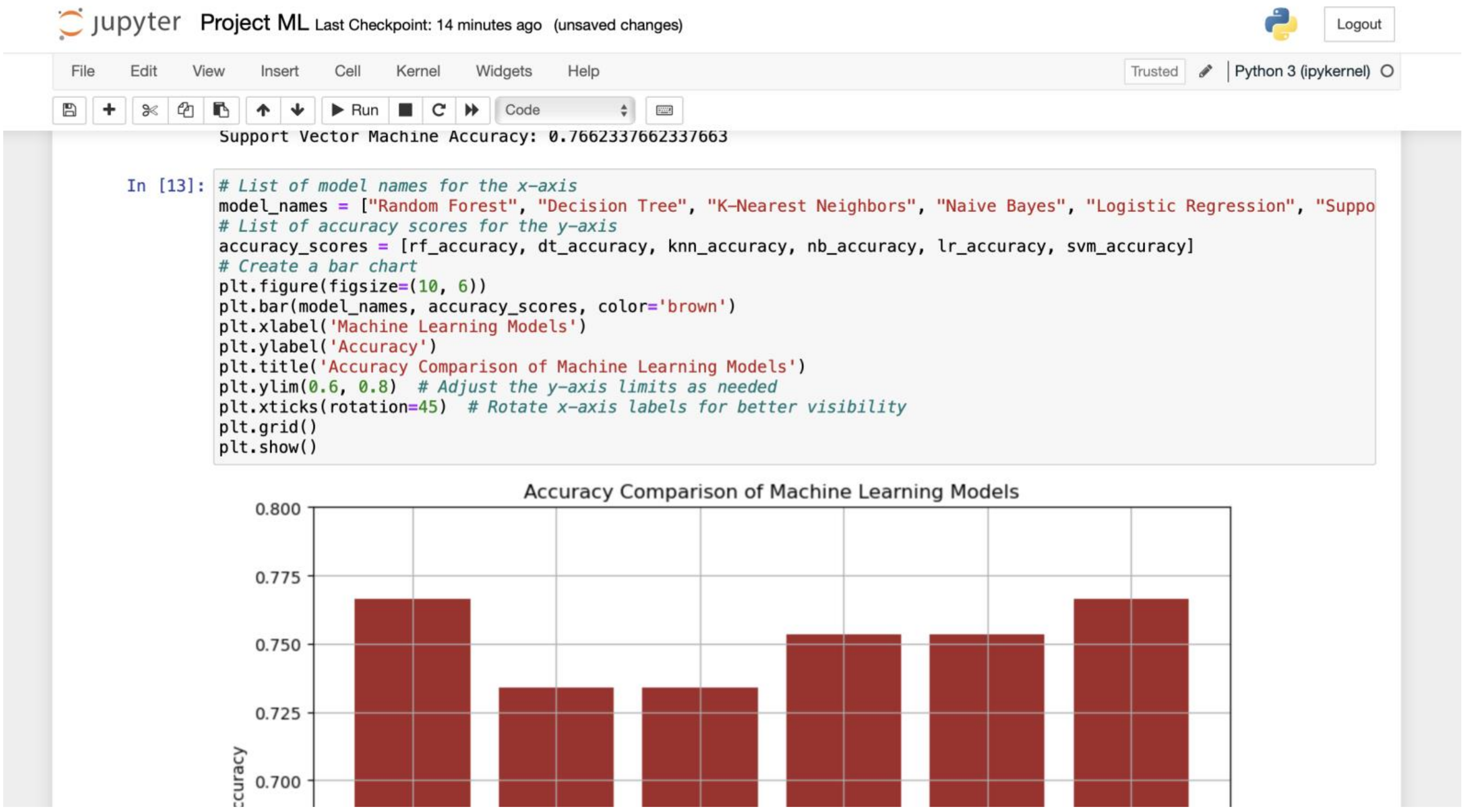
```python
print("Random Forest Accuracy:", rf_accuracy ,"\n Confusion Matrix: \n",rf_conf)

print("Decision Tree Accuracy:", dt_accuracy,"\n Confusion Matrix: \n",dt_conf)
print("K-Nearest Neighbors Accuracy:", knn_accuracy,"\n Confusion Matrix: \n",knn_conf)
print("Naive Bayes Accuracy:", nb_accuracy,"\n Confusion Matrix: \n",nb_conf)
print("Logistic Regression Accuracy:", lr_accuracy,"\n Confusion Matrix: \n",lr_conf)
print("Support Vector Machine Accuracy:", svm_accuracy,"\n Confusion Matrix: \n",svm_conf)
```

```
Random Forest Accuracy: 0.7727272727272727
 Confusion Matrix:
 [[80 19]
 [16 39]]
Decision Tree Accuracy: 0.7077922077922078
 Confusion Matrix:
 [[72 27]
 [18 37]]
K-Nearest Neighbors Accuracy: 0.7337662337662337
 Confusion Matrix:
 [[76 23]
 [18 37]]
Naive Bayes Accuracy: 0.7532467532467533
 Confusion Matrix:
 [[79 20]
 [18 37]]
Logistic Regression Accuracy: 0.7532467532467533
 Confusion Matrix:
 [[81 18]
 [20 35]]
Support Vector Machine Accuracy: 0.7662337662337663
 Confusion Matrix:
 [[82 17]
 [19 36]]
```

Eventually reached to print the accuracy and confusion matrix for each model here as appearing in image we have printed all Accuracy Score and Confusion Matrix

In the beginning , to accurately assess the performance of each model, we need to compare their accuracy visually.

For this, we created a bar chart. Initially, we listed the names of each model in an array, and then we recorded their accuracies in another array. Finally, we formatted the bar chart as depicted in the following image.



```
Support Vector Machine Accuracy: 0.7662337662337663
```

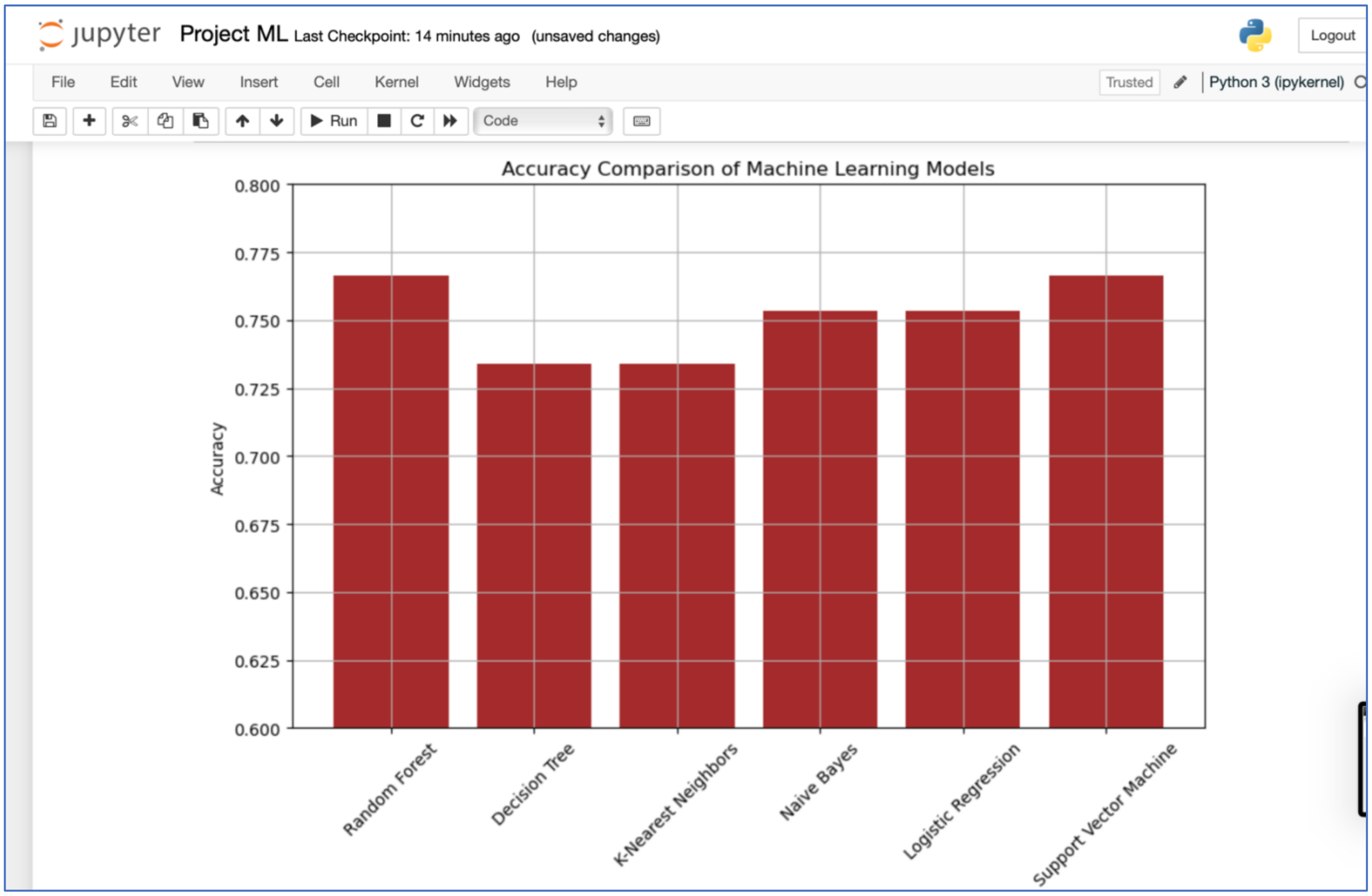
```
In [13]: # List of model names for the x-axis
model_names = ["Random Forest", "Decision Tree", "K-Nearest Neighbors", "Naive Bayes", "Logistic Regression", "Suppo
# List of accuracy scores for the y-axis
accuracy_scores = [rf_accuracy, dt_accuracy, knn_accuracy, nb_accuracy, lr_accuracy, svm_accuracy]
# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, accuracy_scores, color='brown')
plt.xlabel('Machine Learning Models')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison of Machine Learning Models')
plt.ylim(0.6, 0.8)  # Adjust the y-axis limits as needed
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.grid()
plt.show()
```

In the next image, you'll find a comprehensive visualization that provides insights into the accuracy of each model as well as its comparative performance against other models.

This plot offers a clear and intuitive representation, allowing for a detailed examination of each model's accuracy in relation to the others.

By presenting the data in this manner, you can easily discern not only the absolute accuracy of each model but also how it stacks up against its counterparts.

This visual depiction serves as a valuable tool for decision-making, enabling you to identify the most effective model for their specific need and objective



As depicted in previous the plot, it's evident that both SVM and Random Forest algorithms exhibit the highest accuracy among

the various machine learning algorithms considered. Notably, they achieve this superior performance while requiring fewer input data points compared to their counterparts.

This characteristic renders them particularly well-suited for less resource-intensive applications characterized by smaller datasets.

Their ability to deliver exceptional accuracy with minimal input data underscores their versatility and efficacy in scenarios where computational resources are limited or where data collection is constrained.

Therefore, SVM and Random Forest algorithms emerge as optimal choices for such applications, offering a compelling balance between accuracy and resource efficiency

What have we gained from this project?

Throughout the duration of this project, we've embarked on an enriching journey that has significantly augmented our expertise in the realm of machine learning. Our meticulous exploration of various algorithms has not only broadened our understanding of their intricate workings but has also deepened our appreciation for the nuances inherent in selecting the most suitable algorithms for diverse tasks.

As we delved into the intricacies of each algorithm, tirelessly seeking optimal solutions, we've honed our ability to discern which algorithms are best aligned with the characteristics of our dataset. This process has not only allowed us to identify and leverage algorithms that excel in handling our data but has also necessitated the discernment to discard those that do not meet our criteria, thereby reinforcing our understanding of algorithmic principles.

Moreover, our journey has afforded us a profound insight into the inner workings of algorithms, from their fundamental mechanics to the intricacies of training them effectively. We've delved into the nuances of algorithm optimization, exploring techniques to fine-tune their performance and enhance their efficacy in real-world applications.

Additionally, our experience has underscored the paramount importance of data cleanliness as a cornerstone for achieving model accuracy. We've come to recognize that the quality and integrity of our data are foundational elements that underpin the success of our models, emphasizing the significance of meticulous data preprocessing and cleaning practices.

In essence, our project has not only expanded our scientific knowledge but has also equipped us with a repertoire of practical skills essential for navigating the complexities of real-world machine learning challenges. Through our diligent exploration and experimentation, we've emerged with a deeper understanding of machine learning principles and a heightened proficiency in applying them to solve complex problems.

Conclusion :

The project aimed to leverage machine learning techniques for diagnosing patients and detecting diseases, particularly focusing on diabetes. By analyzing extensive medical data including patient history, symptoms, and test results, machine learning algorithms enable healthcare professionals to make more accurate diagnoses and create personalized treatment plans, ultimately improving patient outcomes. Additionally, wearable devices and remote monitoring systems allow for continuous health tracking, facilitating early detection and proactive management of conditions like diabetes.

The dataset used in this project, sourced from Kaggle Platform contains diagnostic measurements aimed at predicting whether a patient has diabetes. Features include pregnancies, glucose levels, blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, and age, with the target variable indicating the final diagnosis (1 for yes, 0 for no).

A meticulous selection process led to the adoption of machine learning algorithms, with Random Forest and SVM standing out for their robust performance in handling complex data relationships and delivering accurate predictions. Complementing these were Logistic Regression, Naive Bayes, and kNN models, ensuring a comprehensive evaluation of accuracy and efficacy.

Implementation was carried out using JupyterLab, providing a versatile and user-friendly environment for model training and experimentation.

To evaluate model performance comprehensively, a detailed visual representation in the form of a bar chart was created. This visualization facilitated a comparative analysis of each model's accuracy, empowering stakeholders to make informed decisions regarding model selection and deployment.

The results revealed that SVM and Random Forest algorithms exhibited the highest accuracy levels while requiring fewer input data points compared to other models. This characteristic renders them suitable for less resource-intensive applications with smaller datasets, offering a compelling balance between accuracy and resource efficiency.

Reference:

You can find the chosen dataset through Kaggle Platform from this link :

https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset/data