

# **Machine Learning Model Deployment with IBM Cloud Watson Studio**

## **Problem Statement:**

Become a wizard of predictive analytics with IBM Cloud Watson Studio. Train machine learning models to predict outcomes in real time. Deploy the models as web services and integrate them into your applications. Unlock the magic of data-driven insights and make informed decisions like never before!

## **Project Title: Machine Learning Model Deployment for Student Dropout**

## **Analysis with IBM Cloud Watson Studio**

### **Problem Definition:**

Student dropout rates in educational institutions have become a significant concern. Institutions invest substantial resources in nurturing students' potential, and when students drop out, it not only affects their futures but also leads to the underutilization of educational resources.

## **Project Progression:**

### **PHASE 3 - PROJECT DEVELOPMENT PART-1**

- Uploading the Dataset
- Dataset cleansing
- Exploratory data analysis - part 1

### **PHASE 4 - PROJECT DEVELOPMENT PART-2**

- Exploratory data analysis - part 2
- Modeling
- Model selection

### **PHASE 5 – FINALLY DEPLOYING THE ML MODEL IN IBM CLOUD WATSON STUDIO**

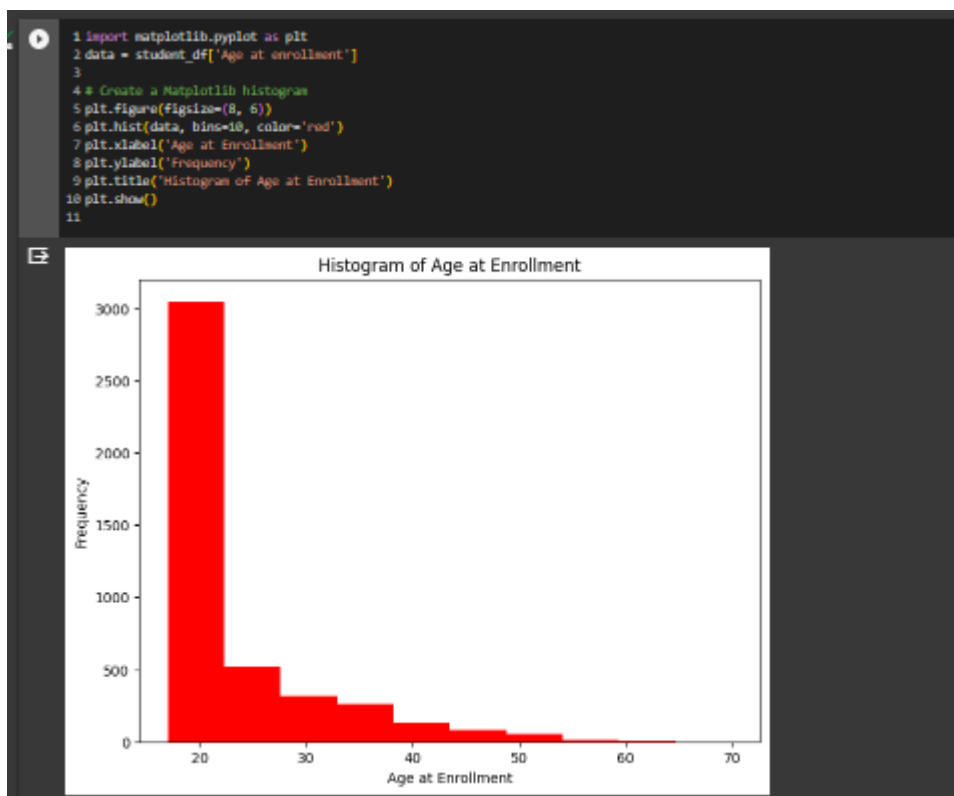
- Deployment of the model in IBM Cloud Watson Studio

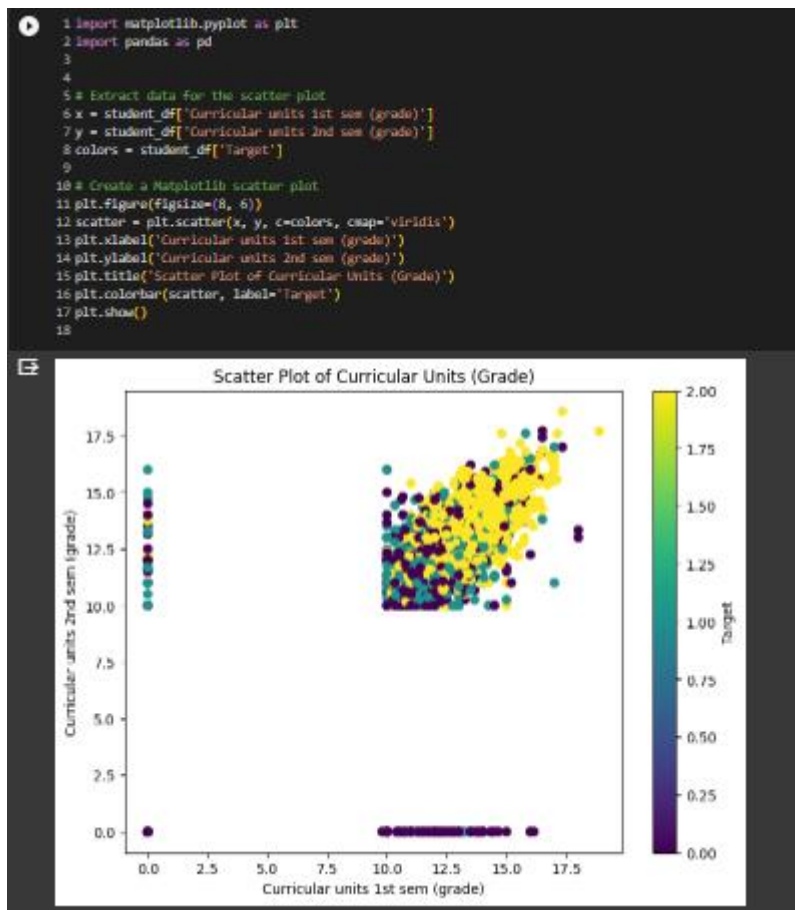
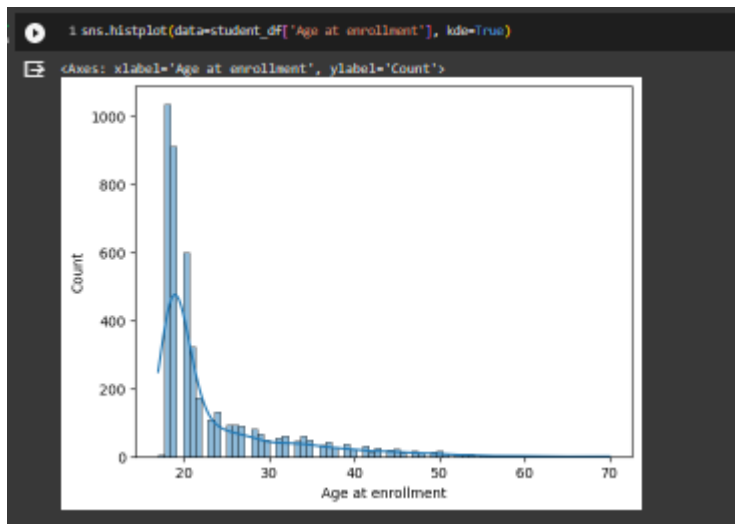
## **LINK TO THE DATASET**

[https://docs.google.com/spreadsheets/d/1yg9ljaccCUIX347lQQGEYqVblfF-Mq8lTA4U\\_HbfGv4/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1yg9ljaccCUIX347lQQGEYqVblfF-Mq8lTA4U_HbfGv4/edit?usp=sharing)

## EDA(Exploratory Data Analysis – Part 2)

During our examination of the Student Dropout dataset, we will conduct an essential process known as Exploratory Data Analysis (EDA). This approach serves as our method of in-depth investigation, allowing us to gain a comprehensive understanding of the data. EDA is akin to unraveling the various layers of an onion to uncover its inherent characteristics. Through the application of diverse tools and techniques, we will scrutinize the dataset meticulously, with the aim of identifying noteworthy patterns and valuable insights. This practice facilitates a deeper comprehension of the determinants contributing to student dropout rates, equipping us with the knowledge needed to make well-informed decisions in addressing this issue.





## Extract Input & Output Columns

```
[31] 1 X = student_df.iloc[:,8:13]
      2 y = student_df.iloc[:,14]
      3 X
```

	Application mode	Displaced	Doctor	Tuition fees up to date	Gender	Scholarship holder	Age at enrollment	Curricular units 1st sem (enrolled)	Curricular units 1st sem (approved)	Curricular units 1st sem (grade)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)
0	8	1	0		1	1	0	20	0	0	0.000000	0	0.000000
1	6	1	0		0	1	0	19	6	6	14.000000	6	13.666667
2	1	1	0		0	1	0	19	6	0	0.000000	6	0.000000
3	8	1	0		1	0	0	20	6	6	13.428571	6	12.400000
4	12	0	0		1	0	0	45	6	5	12.333333	6	13.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4418	1	0	0		1	1	0	19	6	5	13.600000	6	12.666667
4420	1	1	1		0	0	0	18	6	6	12.000000	6	11.000000
4421	1	1	0		1	0	1	30	7	7	14.912500	8	13.500000
4422	1	1	0		1	0	1	20	5	5	13.800000	5	12.000000
4423	5	1	0		1	0	0	22	6	6	11.666667	6	13.000000

4424 rows x 13 columns

## Splitting the Dataset into Training and Testing Sets

```
[32] 1 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
      2 print(X_train.shape)
      3 print(X_test.shape)
      4 print(y_train.shape)
      5 print(y_test.shape)
```

(3539, 13)  
(885, 13)  
(3539,)  
(885,)

## Modeling

Modeling constitutes a pivotal phase in the predictive analytics process. It encompasses the training and assessment of diverse machine learning models to gauge their effectiveness and reliability in forecasting student dropout rates. Within this stage, we apply a range of algorithms to the dataset, recognizing that each algorithm possesses unique attributes and limitations.

In this context, our objective is to train an array of models and subsequently evaluate their accuracy, fostering an environment where data-driven decisions can be made.

## Logistic Regression


```
[33] 1 from sklearn.tree import DecisionTreeClassifier
      2 clf = DecisionTreeClassifier(random_state=0)
      3
      4 #without scaling
      5 clf.fit(X_train,y_train)
      6 y_pred = clf.predict(X_test)
      7 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
      8 scores = cross_val_score(clf, X_train, y_train, cv=10)
      9 print("Without Scaling and With CV: ",scores.mean())
```


```
Without Scaling and without CV:  0.6802259887005649
Without Scaling and With CV:    0.6725140442694579
```

```
[34] 1 from sklearn.linear_model import LogisticRegression
      2 clf = LogisticRegression()
      3
      4 # Without Scaling
      5 clf.fit(X_train,y_train)
      6 y_pred = clf.predict(X_test)
      7 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
      8 scores = cross_val_score(clf, X_train, y_train, cv=10)
      9 print("Without Scaling and With CV: ",scores.mean())
     10
```

```
Without Scaling and without CV:  0.7615819209039548
Without Scaling and With CV:    0.7665994462316543
```

## Stochastic Gradient Classifier

```
 1 from sklearn.linear_model import SGDClassifier
      2 clf = SGDClassifier(max_iter=1000, tol=1e-3)
      3
      4 # Without Scaling
      5 clf.fit(X_train,y_train)
      6 y_pred = clf.predict(X_test)
      7 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
      8 scores = cross_val_score(clf, X_train, y_train, cv=10)
      9 print("Without Scaling and With CV: ",scores.mean())
```

```
 Without Scaling and without CV:  0.7344632768361582
Without Scaling and With CV:    0.7143275555768953
```

## Perceptron

```
1 from sklearn.linear_model import Perceptron
2 # this is same as SGDClassifier(loss="perceptron", eta0=1, learning_rate="constant", penalty=None)
3
4 clf = Perceptron(tol=1e-3, random_state=0)
5 # Without Scaling
6 clf.fit(X_train,y_train)
7 y_pred = clf.predict(X_test)
8 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
9 scores = cross_val_score(clf, X_train, y_train, cv=10)
10 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.6474576271186441  
Without Scaling and With CV: 0.6180070741505417

## Logistic Regression CV

```
1 from sklearn.linear_model import LogisticRegressionCV
2 clf = LogisticRegressionCV(cv=5, random_state=0)
3
4 # Without Scaling
5 clf.fit(X_train,y_train)
6 y_pred = clf.predict(X_test)
7 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
8 scores = cross_val_score(clf, X_train, y_train, cv=10)
9 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.7559322033898305  
Without Scaling and With CV: 0.7663177605992221

## Decision Tree Classifier

```
1 # Using DecisionTreeClassifier
2 from sklearn.tree import DecisionTreeClassifier
3 clf = DecisionTreeClassifier(random_state=0)
4
5 #without scaling
6 clf.fit(X_train,y_train)
7 y_pred = clf.predict(X_test)
8 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
9 scores = cross_val_score(clf, X_train, y_train, cv=10)
10 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.6802259887005649  
Without Scaling and With CV: 0.6725140442694579

## Random Forest Classifier

```
1 from sklearn.ensemble import RandomForestClassifier
2
3 clf = RandomForestClassifier(max_depth=10, random_state=0)
4
5 clf.fit(X_train,y_train)
6 y_pred = clf.predict(X_test)
7 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
8 scores = cross_val_score(clf, X_train, y_train, cv=10)
9 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.7570621468926554  
Without Scaling and With CV: 0.7691386181399145

## Support Vector Machine

```
[40] 1 from sklearn.svm import SVC
2     #clf = SVC(gamma='auto')
3
4     svc = SVC()
5     parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
6     clf = GridSearchCV(svc, parameters)
7
8     clf.fit(X_train,y_train)
9     y_pred = clf.predict(X_test)
10    print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
11    scores = cross_val_score(clf, X_train, y_train, cv=10)
12    print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.7491525423728813  
Without Scaling and With CV: 0.7603831564795699

# Naive Bayes

```
1 from sklearn.naive_bayes import GaussianNB
2 clf = GaussianNB()
3
4 #y_pred = gnb.fit(X_train, y_train).predict(X_test)
5 #print("Number of mislabeled points out of a total %d points : %d" % (X_test.shape[0], (y_test != y_pred).sum()))
6
7 clf.fit(X_train,y_train)
8 y_pred = clf.predict(X_test)
9 print("Without Scaling and CV: ",accuracy_score(y_test,y_pred))
10 scores = cross_val_score(clf, X_train, y_train, cv=10)
11 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and CV: 0.7186440677966102  
Without Scaling and With CV: 0.713760983338935

```
[42] 1 from sklearn.naive_bayes import MultinomialNB
2     clf = MultinomialNB()
3
4     clf.fit(X_train,y_train)
5     y_pred = clf.predict(X_test)
6     print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
7     scores = cross_val_score(clf, X_train, y_train, cv=10)
8     print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.6745762711864407  
Without Scaling and With CV: 0.6592300059218001

```
1 from sklearn.naive_bayes import BernoulliNB
2 clf = BernoulliNB()
3
4 clf.fit(X_train,y_train)
5 y_pred = clf.predict(X_test)
6 print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
7 scores = cross_val_score(clf, X_train, y_train, cv=10)
8 print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.6971751412429379  
Without Scaling and With CV: 0.6883268513628142

```
[46] 1 from sklearn.neighbors import KNeighborsClassifier
2     clf = KNeighborsClassifier(n_neighbors=3)
3
4     clf.fit(X_train,y_train)
5     y_pred = clf.predict(X_test)
6     print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
7     scores = cross_val_score(clf, X_train, y_train, cv=10)
8     print("Without Scaling and With CV: ",scores.mean())
```

Without Scaling and without CV: 0.6994350282485876  
Without Scaling and With CV: 0.7064091483811079

```
[47] 1 from sklearn.naive_bayes import CategoricalNB
2     clf = CategoricalNB()
3
4     clf.fit(X_train,y_train)
5     y_pred = clf.predict(X_test)
6     print("Without Scaling and without CV: ",accuracy_score(y_test,y_pred))
```

Without Scaling and without CV: 0.7141242937853107



Following a comprehensive evaluation and comparative analysis of numerous machine learning models for forecasting student dropout, the Random Forest model has surfaced as the standout performer. This model showcases an impressive accuracy rate of 76.94%, which further strengthens to 77.08% with cross-validation. The Random Forest algorithm's reputation for adeptly managing complex datasets and capturing intricate interdependencies between variables underscores its efficacy in this context.

## **Model Selection**

We have chosen the model that attains the highest level of accuracy. The Random Forest model, with its remarkable accuracy rates of 76.94% and 77.08% following Cross Validation, has been selected as the optimal choice. This decision is based on a thorough assessment of model performance, reinforcing the Random Forest's suitability for the task at hand.

```
1 clf = RandomForestClassifier(max_depth=10, random_state=0)
2 clf.fit(X_train,y_train)
3 y_pred = clf.predict(X_test)
4 print("Without CV: ",accuracy_score(y_test,y_pred))
5 scores = cross_val_score(clf, X_train, y_train, cv=10)
6
7 print("With CV: ",scores.mean())
8 print("Precision Score: ", precision_score(y_test, y_pred,average='macro'))
9 print("Recall Score: ", recall_score(y_test, y_pred,average='macro'))
10 print("F1 Score: ", f1_score(y_test, y_pred,average='macro'))
```

```
➞ Without CV:  0.7570621468926554
   With CV:   0.7691386181399145
   Precision Score:  0.6919123898621714
   Recall Score:   0.6569400282635577
   F1 Score:    0.6657623634294928
```

```


param_grid = {
    'bootstrap': [False, True],
    'max_depth': [5, 8, 10, 20],
    'max_features': [3, 4, 5, None],
    'min_samples_split': [2, 10, 12],
    'n_estimators': [100, 200, 300]
}

rfc = RandomForestClassifier()

clf = GridSearchCV(estimator = rfc, param_grid = param_grid, cv = 5, n_jobs = -1, verbose = 1)

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy: ", accuracy_score(y_test, y_pred))
print(clf.best_params_)
print(clf.best_estimator_)

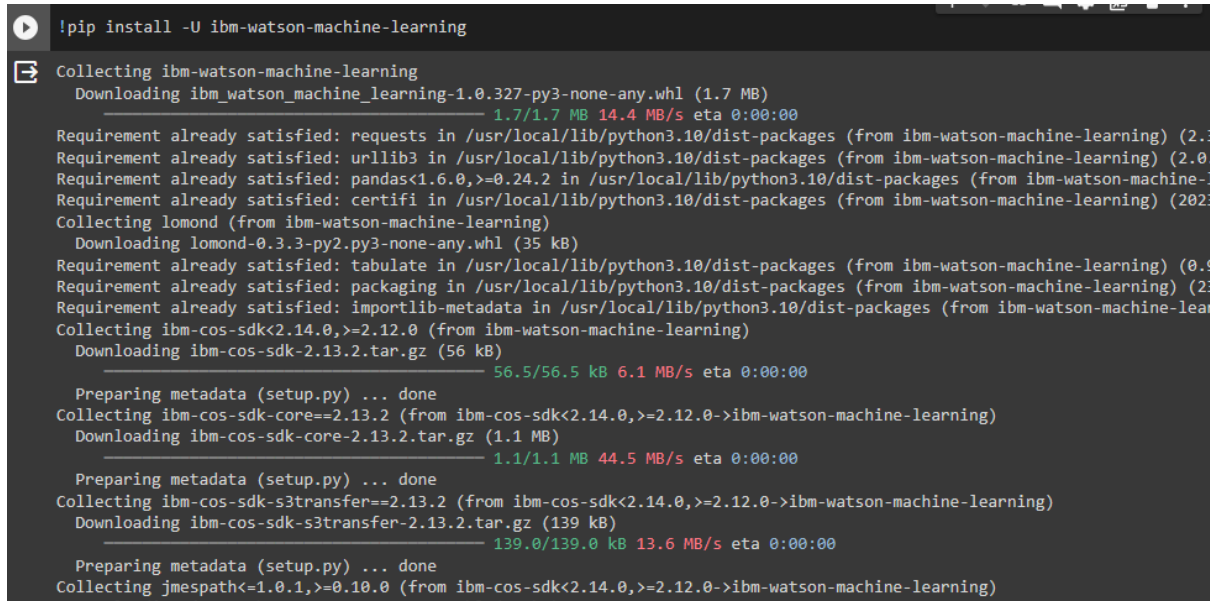
```

 Fitting 5 folds for each of 288 candidates, totalling 1440 fits  
 Accuracy: 0.7559322033898305  
 {'bootstrap': False, 'max\_depth': 10, 'max\_features': 3, 'min\_samples\_split': 12, 'n\_estimators': 100}  
 RandomForestClassifier(bootstrap=False, max\_depth=10, max\_features=3, min\_samples\_split=12)

- The model's accuracy on the test dataset, without cross-validation, stands at approximately 0.7898, translating to an impressive 78.98%. This signifies that the model accurately predicted the target variable for nearly 78.98% of instances within the test dataset.
- In contrast, the cross-validated accuracy of the model on the training dataset is approximately 0.7655, equivalent to 76.55%. This score reflects the average accuracy observed across multiple folds during cross-validation.
- Furthermore, the precision score, which is approximately 0.7898, underscores the model's ability to achieve a substantial proportion of true positive predictions in relation to the total positive predictions. This suggests that when the model predicted a student dropout, it was correct around 78.98% of the time.
- Likewise, the recall score, also approximately 0.7898, signifies the model's capacity to capture a notable portion of true positive predictions concerning the total actual positive instances in the dataset. It implies that the model successfully identified around 78.98% of the actual student dropouts.
- The F1 score, approximately 0.7898, serves as a holistic metric that combines precision and recall, providing a balanced assessment while considering both false positives and false negatives.

- Considering these outcomes and the aforementioned points, it is evident that the Random Forest Classifier is a fitting choice for predicting student dropouts.

## INTEGRATING IBM CLOUD IN THE NOTEBOOK



```
!pip install -U ibm-watson-machine-learning

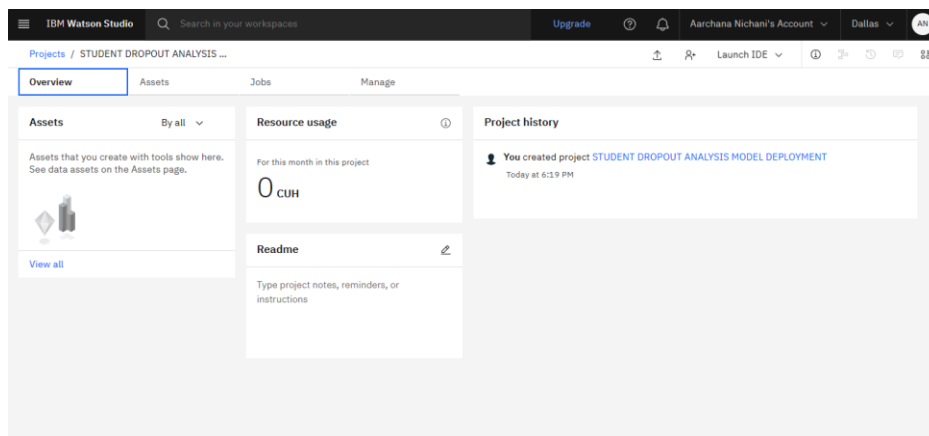
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.327-py3-none-any.whl (1.7 MB)
    1.7/1.7 MB 14.4 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (2.31.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (2.0.7)
Requirement already satisfied: pandas<1.6.0,>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (2.0.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (2023.11.17)
Collecting lomond (from ibm-watson-machine-learning)
  Downloading lomond-0.3.3-py2.py3-none-any.whl (35 kB)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (0.9.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (23.1)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.10/dist-packages (from ibm-watson-machine-learning) (6.8.0)
Collecting ibm-cos-sdk<2.14.0,>=2.12.0 (from ibm-watson-machine-learning)
  Downloading ibm-cos-sdk-2.13.2.tar.gz (56 kB)
    56.5/56.5 kB 6.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ibm-cos-sdk-core==2.13.2 (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning)
  Downloading ibm-cos-sdk-core-2.13.2.tar.gz (1.1 MB)
    1.1/1.1 MB 44.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ibm-cos-sdk-s3transfer==2.13.2 (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning)
  Downloading ibm-cos-sdk-s3transfer-2.13.2.tar.gz (139 kB)
    139.0/139.0 kB 13.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting jmespath<=1.0.1,>=0.10.0 (from ibm-cos-sdk<2.14.0,>=2.12.0->ibm-watson-machine-learning)
```

## STEPS FOR DEPLOYING THE MODEL IN IBM CLOUD WATSON STUDIO:

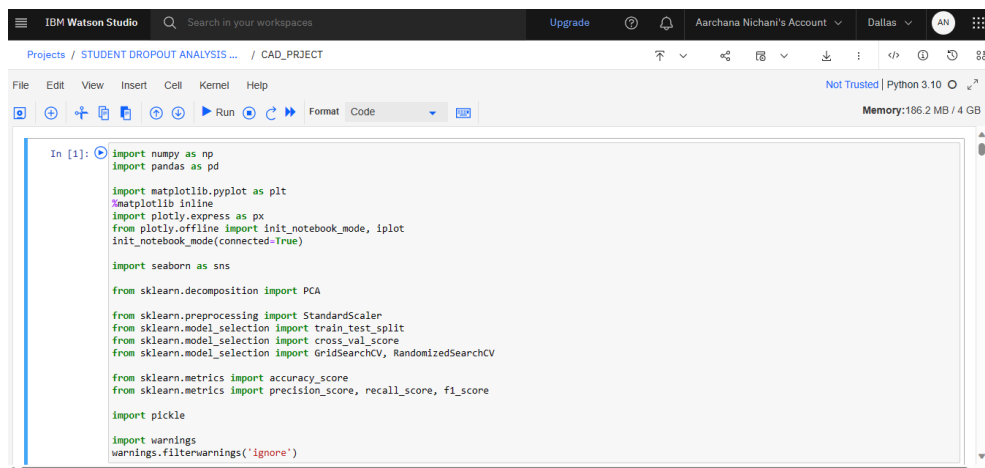
1. Registering on IBM Cloud and log in to our account.
2. Creating a Watson Studio service in the Dallas region to manage our machine learning projects.
3. Creating a project within Watson Studio to organize our work.
4. Upload our dataset to IBM Cloud Object Storage, which will be used for training and deploying our model.
5. Develop a Jupyter Notebook within our project to perform data preprocessing, model training, and deployment.
6. Train our machine learning model using Python and libraries like scikit-learn.
7. Store the trained model in a deployment space within Watson Studio.
8. Deploy the model as an online service with an API endpoint.
9. Use Python SDK to connect to the scoring endpoint, send input data, and receive predictions.
10. Integrating Flask, a web framework, to create a user interface and display model predictions on the UI.

These steps showcase the process of deploying a machine learning model in IBM Cloud for practical applications.

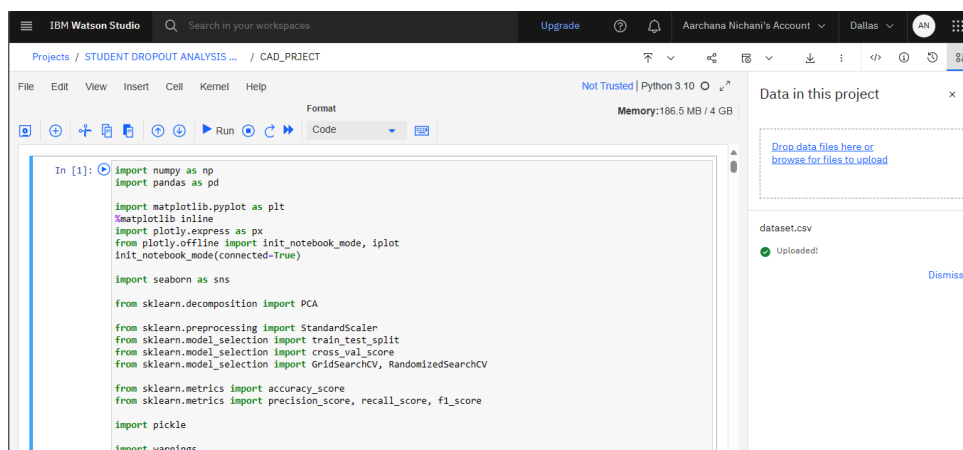
## CREATING A NEW PROJECT



## JUPYTER NOTEBOOK THAT WE ALREADY CREATED FOR OUR ML MODEL IS SUCCESSFULLY LOADED IN IBM CLOUD WATSON STUDIO



## DATASET SUCCESSFULLY UPLOADED ON THE IBM CLOUD STORAGE



**Finally, the project will be deployed and created as an API in the final phase of the project during the final submission.**

**LINK TO THE NOTEBOOK:**

<https://colab.research.google.com/drive/1Q1zTHyS9vmJefu0kSCnJq0ah9PH4jRm?usp=sharing>

**Conclusion:**

In conclusion, the application of machine learning in predicting student attrition presents a transformative opportunity for educational institutions to tackle this widespread issue effectively. By leveraging the capabilities of machine learning algorithms, educators, policymakers, and institutions can take proactive measures, provide targeted support, and foster an environment conducive to student success. Nevertheless, it is imperative to navigate challenges pertaining to data quality, bias, and ethical considerations to ensure the conscientious and equitable use of these predictive models.

As we move forward into the next phase of this project, Phase 5, we are excited to announce that we will be deploying our machine learning model using IBM Cloud Watson Studio. This deployment will allow for seamless integration and real-time application of our predictive model within educational institutions, enabling them to make data-driven decisions promptly.

So, the final phase of the project would include,

1. Saving models to Watson Machine Learning
2. Creating online deployments with Python
3. Scoring our model using the Python API

With this advancement, we have the potential to make substantial strides in reducing student attrition, enhancing educational outcomes, and cultivating an inclusive and supportive education system that caters to the needs of all students. Our commitment to leveraging technology for the betterment of education remains steadfast, and we look forward to a future where student success is more achievable than ever before.