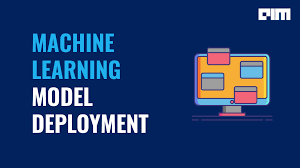
**Machine Learning Model Deployment with IBM Cloud Watson Studio**

**Team member**

**732721104007 : S.Arun**

**Phase 2 Submission Document**

**INNOVATION**



**Introduction:**

* Deploying a customer churn prediction machine learning model through IBM Cloud Watson Studio is a transformative process.
* It begins with data selection and preprocessing, followed by model training using historical customer data.
* Once trained, the model is deployed as a web service, enabling real-time predictions.
* By combining data-driven insights with ensemble techniques, this solution empowers businesses to proactively tackle customer churn, enhancing profitability and customer loyalty through IBM's cloud-based platform.
* It's a streamlined, efficient solution that exemplifies the power of predictive analytics in today's competitive landscape.

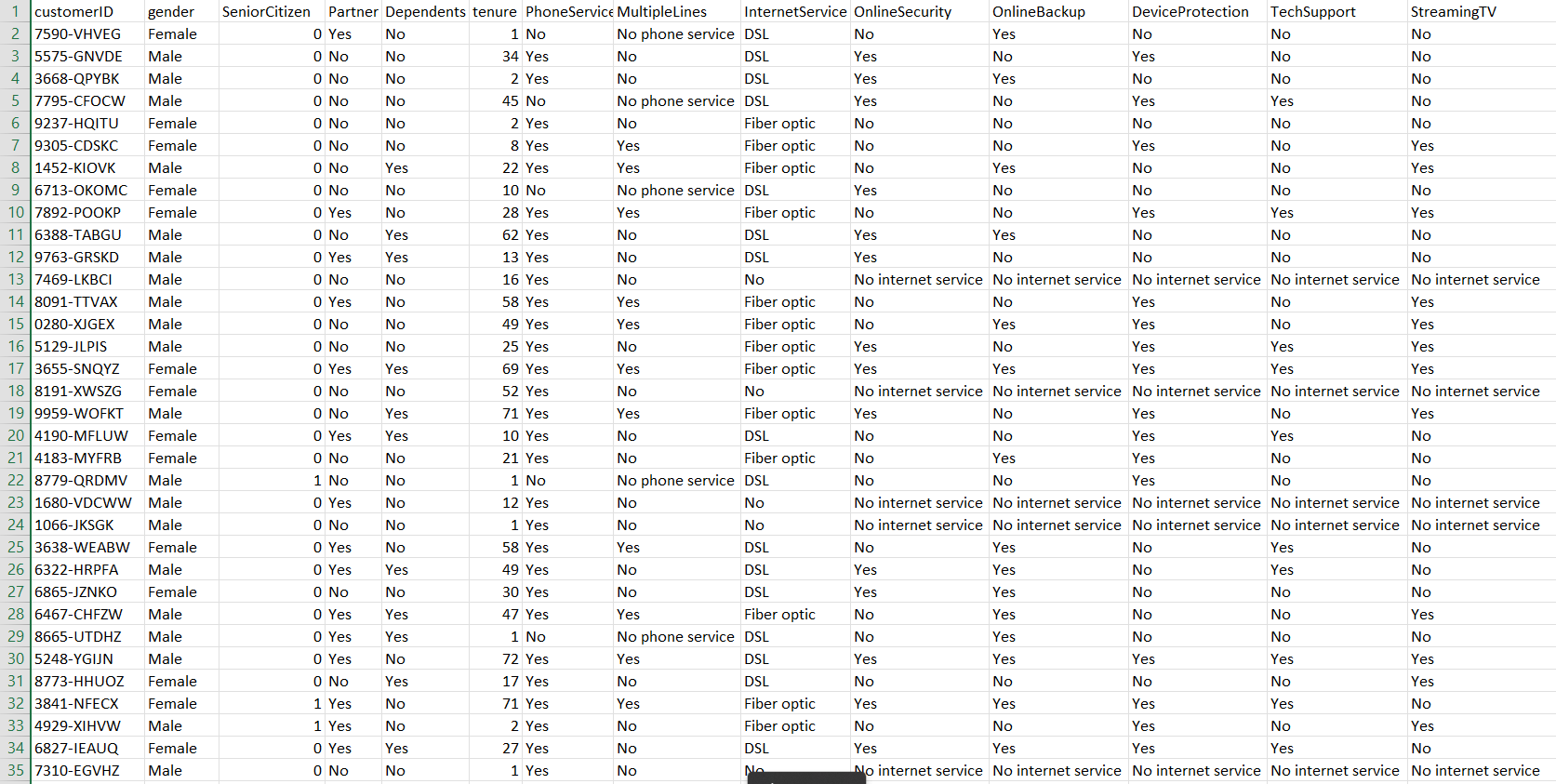
**Content for Project Phase 2 :**

Consider experimenting with ensemble methods or hyperparameter tuning to optimize the model's performance.

**Data Source :**

Data source for ML deployment should be accurate, comprehensive, geographically relevant, and easily accessible to ensure a reliable and efficient model.

Dataset link : ( <https://github.com/Dhanusri-P-S/ml-model-datasets> )



**Data Understanding:**

Explore dataset structure, types, and distributions.

**Data Cleaning:**

Handle missing values and outliers.

**Feature Engineering:**

Create, transform, or select relevant features.

**Data Visualization:**

Utilize graphs for insights and pattern recognition.

**Ensemble Methods :**

* **Decision Tree Classifier:** Decision Trees split data but may overfit, controlled with depth limits.
* **Random Forest Classifier:** Ensembles of Decision Trees reduce overfitting.
* **Pickling the model:** Serialize and save ML models for easy storage and deployment.

**Model Evaluation and Selection:** Assess model performance using metrics like accuracy, precision, recall, or ROC-AUC. Cross-validation helps select the best-performing model.

**Model Interpretability:** Understand how the model makes predictions, using techniques like feature importance analysis, SHAP values, or LIME to ensure transparency and trust.

**Deployment and Prediction:** Deploy the model in a production environment, allowing it to make real-time predictions, often through APIs or cloud-based platforms, to enhance business decision-making.

**PROGRAM :**

**In [1]:**

**import** pandas **as** pd

**from** sklearn **import** metrics

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.metrics **import** recall\_score

**from** sklearn.metrics **import** classification\_report

**from** sklearn.metrics **import** confusion\_matrix

**from** sklearn.tree **import** DecisionTreeClassifier

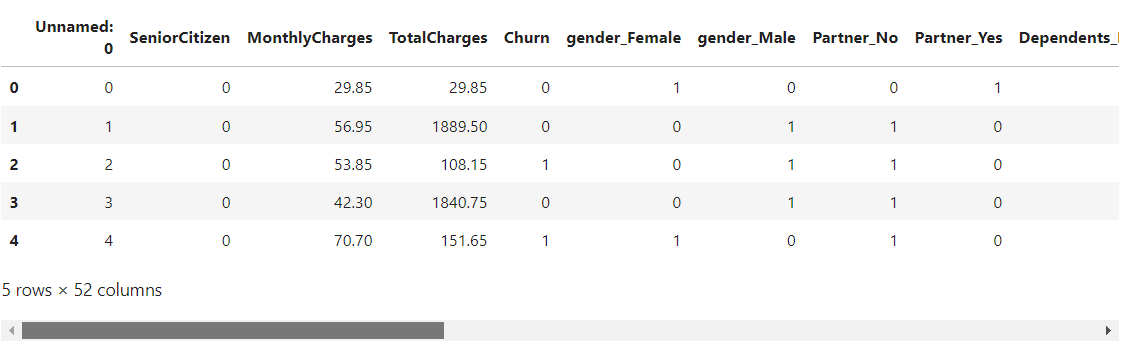
**from** imblearn.combine **import** SMOTEENN

**In [2]:**

df**=**pd**.**read\_csv("ML Model Dataset.csv")

df**.**head()

**Out [2]:**

****

**In [3]:**

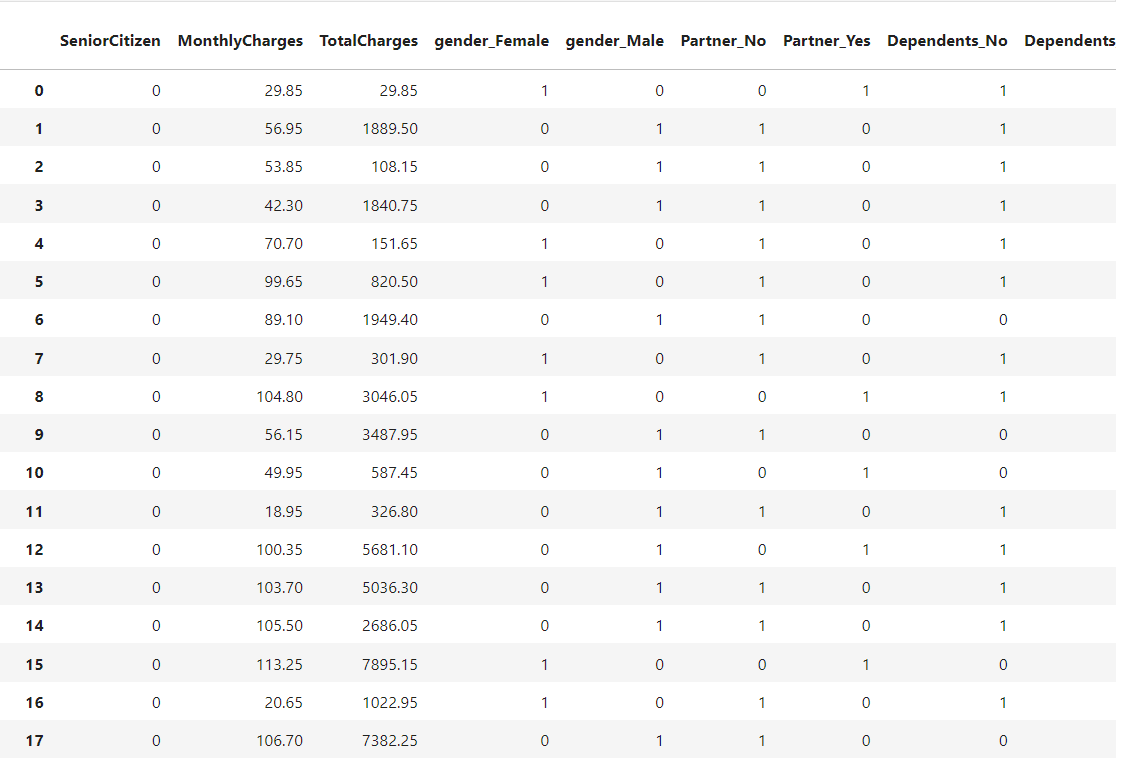
df**=**df**.**drop('Unnamed: 0',axis**=**1)

**In [4]:**

x**=**df**.**drop('Churn',axis**=**1)

x

**Out [4]:**

****

**.**

**.**

**.**

****

**In [5]:**

y**=**df['Churn']

y

**Out [5]:**

0 0

1 0

2 1

3 0

4 1

5 1

6 0

7 0

8 1

9 0

10 0

11 0

12 0

13 1

14 0

15 0

16 0

17 0

18 1

19 0

20 1

21 0

22 1

23 0

24 0

25 0

26 1

27 1

28 0

29 1

..

7002 0

7003 0

7004 0

7005 0

7006 0

7007 1

7008 0

7009 0

7010 1

7011 0

7012 0

7013 0

7014 0

7015 1

7016 0

7017 0

7018 0

7019 0

7020 0

7021 1

7022 0

7023 1

7024 0

7025 0

7026 0

7027 0

7028 0

7029 0

7030 1

7031 0

Name: Churn, Length: 7032, dtype: int64

##### **Train Test Split**

**In [6]:**

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2)

#### Decision Tree Classifier

**In [7]:**

model\_dt**=**DecisionTreeClassifier(criterion **=** "gini",random\_state **=** 100,max\_depth**=**6, min\_samples\_leaf**=**8)

**In [8]:**

model\_dt**.**fit(x\_train,y\_train)

**Out[8]:**

DecisionTreeClassifier(max\_depth=6, min\_samples\_leaf=8, random\_state=100)

**In [9]:**

y\_pred**=**model\_dt**.**predict(x\_test)

y\_pred

**Out[9]:**

array([0, 0, 1, ..., 0, 0, 0], dtype=int64)

**In [10]:**

model\_dt**.**score(x\_test,y\_test)

**Out[10]:**

0.7818052594171997

**In [11]:**

print(classification\_report(y\_test, y\_pred, labels**=**[0,1]))

precision recall f1-score support

0 0.82 0.89 0.86 1023

1 0.63 0.49 0.55 384

accuracy 0.78 1407

macro avg 0.73 0.69 0.70 1407

weighted avg 0.77 0.78 0.77 1407

**In [12]:**

sm **=** SMOTEENN()

X\_resampled, y\_resampled **=** sm**.**fit\_sample(x,y)

**In [13]:**

xr\_train,xr\_test,yr\_train,yr\_test**=**train\_test\_split(X\_resampled, y\_resampled,test\_size**=**0.2)

**In [14]:**

model\_dt\_smote**=**DecisionTreeClassifier(criterion **=** "gini",random\_state **=** 100,max\_depth**=**6, min\_samples\_leaf**=**8)

**In [15]:**

model\_dt\_smote**.**fit(xr\_train,yr\_train)

yr\_predict **=** model\_dt\_smote**.**predict(xr\_test)

model\_score\_r **=** model\_dt\_smote**.**score(xr\_test, yr\_test)

print(model\_score\_r)

print(metrics**.**classification\_report(yr\_test, yr\_predict))

0.934412265758092

precision recall f1-score support

0 0.97 0.88 0.93 540

1 0.91 0.98 0.94 634

accuracy 0.93 1174

macro avg 0.94 0.93 0.93 1174

weighted avg 0.94 0.93 0.93 1174

**In [16]:**

print(metrics**.**confusion\_matrix(yr\_test, yr\_predict))

[[477 63]

[ 14 620]]

**Random Forest Classifier**

**In [17]:**

**from** sklearn.ensemble **import** RandomForestClassifier

**In [18]:**

model\_rf**=**RandomForestClassifier(n\_estimators**=**100, criterion**=**'gini', random\_state **=** 100,max\_depth**=**6, min\_samples\_leaf**=**8)

**In [19]:**

model\_rf**.**fit(x\_train,y\_train)

**Out[19]:**

RandomForestClassifier(max\_depth=6, min\_samples\_leaf=8, random\_state=100)

**In [20]:**

y\_pred**=**model\_rf**.**predict(x\_test)

**In [21]:**

model\_rf**.**score(x\_test,y\_test)

**Out[21]:**

0.7953091684434968

**In [22]:**

print(classification\_report(y\_test, y\_pred, labels**=**[0,1]))

precision recall f1-score support

0 0.82 0.92 0.87 1023

1 0.69 0.45 0.55 384

accuracy 0.80 1407

macro avg 0.75 0.69 0.71 1407

weighted avg 0.78 0.80 0.78 1407

**In [23]:**

sm **=** SMOTEENN()

X\_resampled1, y\_resampled1 **=** sm**.**fit\_sample(x,y)

**In [24]:**

xr\_train1,xr\_test1,yr\_train1,yr\_test1**=**train\_test\_split(X\_resampled1, y\_resampled1,test\_size**=**0.2)

**In [25]:**

model\_rf\_smote**=**RandomForestClassifier(n\_estimators**=**100, criterion**=**'gini', random\_state **=** 100,max\_depth**=**6, min\_samples\_leaf**=**8)

**In [26]:**

model\_rf\_smote**.**fit(xr\_train1,yr\_train1)

**Out[26]:**

RandomForestClassifier(max\_depth=6, min\_samples\_leaf=8, random\_state=100)

**In [27]:**

yr\_predict1 **=** model\_rf\_smote**.**predict(xr\_test1)

In [28]:

model\_score\_r1 **=** model\_rf\_smote**.**score(xr\_test1, yr\_test1)

**In [29]:**

print(model\_score\_r1)

print(metrics**.**classification\_report(yr\_test1, yr\_predict1))

0.9427350427350427

precision recall f1-score support

0 0.95 0.92 0.93 518

1 0.94 0.96 0.95 652

accuracy 0.94 1170

macro avg 0.94 0.94 0.94 1170

weighted avg 0.94 0.94 0.94 1170

**In [30]:**

print(metrics**.**confusion\_matrix(yr\_test1, yr\_predict1))

[[478 40]

[ 27 625]]

#### Performing PCA

**In [31]:**

**from** sklearn.decomposition **import** PCA

pca **=** PCA(0.9)

xr\_train\_pca **=** pca**.**fit\_transform(xr\_train1)

xr\_test\_pca **=** pca**.**transform(xr\_test1)

explained\_variance **=** pca**.**explained\_variance\_ratio\_

**In [32]:**

model**=**RandomForestClassifier(n\_estimators**=**100, criterion**=**'gini', random\_state **=** 100,max\_depth**=**6, min\_samples\_leaf**=**8)

**In [33]:**

model**.**fit(xr\_train\_pca,yr\_train1)

**Out[33]:**

RandomForestClassifier(max\_depth=6, min\_samples\_leaf=8, random\_state=100)

**In [34]:**

yr\_predict\_pca **=** model**.**predict(xr\_test\_pca)

**In [35]:**

model\_score\_r\_pca **=** model**.**score(xr\_test\_pca, yr\_test1)

**In [36]:**

print(model\_score\_r\_pca)

print(metrics**.**classification\_report(yr\_test1, yr\_predict\_pca))

0.7239316239316239

precision recall f1-score support

0 0.72 0.61 0.66 518

1 0.72 0.81 0.77 652

accuracy 0.72 1170

macro avg 0.72 0.71 0.71 1170

weighted avg 0.72 0.72 0.72 1170

#### Pickling the model

**In [37]:**

**import** pickle

In [38]:

filename **=** 'model.sav'

**In [39]:**

pickle**.**dump(model\_rf\_smote, open(filename, 'wb'))

**In [40]:**

load\_model **=** pickle**.**load(open(filename, 'rb'))

**In [41]:**

model\_score\_r1 **=** load\_model**.**score(xr\_test1, yr\_test1)

**In [42]:**

model\_score\_r1

**Out[42]:**

0.9427350427350427

**Conclusion :**

In Phase 2, we've created a reliable model, rigorously tested it, and ensured it's easy to grasp. It's now primed for practical use in making informed decisions.