**Naan Mudhalvan**

**Data Analytics using IBM Cognos**

Phase – 4 (Development Part 2)

**PROGRAM :**

***# Visualize Churn Rate by Internet Services***

plot\_by\_internet\_service **=** churn\_dataset**.**groupby('InternetService')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Bar(

x**=**plot\_by\_internet\_service['InternetService'],

y**=**plot\_by\_internet\_service['Churn'],

width **=** [0.3, 0.3, 0.3],

marker**=**dict(

color**=**['orange', 'green', 'teal'])

)

]

plot\_layout **=** go**.**Layout(

xaxis**=**{"type": "category"},

yaxis**=**{"title": "Churn Rate"},

title**=**'Churn Rate by Internet Service',

plot\_bgcolor **=** 'rgb(243,243,243)',

paper\_bgcolor **=** 'rgb(243,243,243)',

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

***# Visualize Churn Rate by Contract Duration***

plot\_by\_contract **=** churn\_dataset**.**groupby('Contract')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Bar(

x**=**plot\_by\_contract['Contract'],

y**=**plot\_by\_contract['Churn'],

width **=** [0.3, 0.3,0.3],

marker**=**dict(

color**=**['orange', 'green','teal'])

)

]

plot\_layout **=** go**.**Layout(

xaxis**=**{"type": "category"},

yaxis**=**{"title": "Churn Rate"},

title**=**'Churn Rate by Contract Duration',

plot\_bgcolor **=** 'rgb(243,243,243)',

paper\_bgcolor **=** 'rgb(243,243,243)',

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

***# Visualize Relation between Tenure & Churn rate***

plot\_by\_tenure **=** churn\_dataset**.**groupby('tenure')**.**Churn**.**mean()**.**reset\_index()

plot\_data **=** [

go**.**Scatter(

x**=**plot\_by\_tenure['tenure'],

y**=**plot\_by\_tenure['Churn'],

mode**=**'markers',

name**=**'Low',

marker**=** dict(size**=** 5,

line**=** dict(width**=**0.8),

color**=** 'green'

),

)

]

plot\_layout **=** go**.**Layout(

yaxis**=** {'title': "Churn Rate"},

xaxis**=** {'title': "Tenure"},

title**=**'Relation between Tenure & Churn rate',

plot\_bgcolor **=** "rgb(243,243,243)",

paper\_bgcolor **=** "rgb(243,243,243)",

)

fig **=** go**.**Figure(data**=**plot\_data, layout**=**plot\_layout)

po**.**iplot(fig)

***#Perform One Hot Encoding using get\_dummies method***

churn\_dataset **=** pd**.**get\_dummies(churn\_dataset, columns **=** ['Contract','Dependents','DeviceProtection','gender',

'InternetService','MultipleLines','OnlineBackup',

'OnlineSecurity','PaperlessBilling','Partner',

'PaymentMethod','PhoneService','SeniorCitizen',

'StreamingMovies','StreamingTV','TechSupport'],

drop\_first**=True**)

***#Perform Feature Scaling and One Hot Encoding***

**from** sklearn.preprocessing **import** StandardScaler

**#Perform *Feature Scaling on 'tenure', 'MonthlyCharges', 'TotalCharges' in order to bring them on same scale***

standardScaler **=** StandardScaler()

columns\_for\_ft\_scaling **=** ['tenure', 'MonthlyCharges', 'TotalCharges']

***#Apply the feature scaling operation on dataset using fit\_transform() method***

churn\_dataset[columns\_for\_ft\_scaling] **=** standardScaler**.**fit\_transform(churn\_dataset[columns\_for\_ft\_scaling])

***# See subset of values***

churn\_dataset**.**head()

***#Number of columns increased and have suffixes attached, as a result of get\_dummies method.***

churn\_dataset**.**columns

***#Create Feature variable X and Target variable y***

y **=** churn\_dataset['Churn']

X **=** churn\_dataset**.**drop(['Churn','customerID'], axis **=** 1)

***#Split the data into training set (70%) and test set (30%)***

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

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.30, random\_state **=** 50)

***# Machine Learning classification model libraries***

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.svm **import** SVC

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

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

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

***#Fit the logistic Regression Model***

logmodel **=** LogisticRegression(random\_state**=**50)

logmodel**.**fit(X\_train,y\_train)

***#Predict the value for new, unseen data***

pred **=** logmodel**.**predict(X\_test)

***# Find Accuracy using accuracy\_score method***

logmodel\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, pred) **\*** 100, 2)

***#Fit the Support Vector Machine Model***

svcmodel **=** SVC(kernel**=**'linear', random\_state**=**50, probability**=True**)

svcmodel**.**fit(X\_train,y\_train)

***#Predict the value for new, unseen data***

svc\_pred **=** svcmodel**.**predict(X\_test)

***# Find Accuracy using accuracy\_score method***

svc\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, svc\_pred) **\*** 100, 2)

***#Fit the K-Nearest Neighbor Model***

**from** sklearn.neighbors **import** KNeighborsClassifier

knnmodel **=** KNeighborsClassifier(n\_neighbors**=**5, metric**=**'minkowski', p**=**2) *#p=2 represents Euclidean distance, p=1 represents Manhattan Distance*

knnmodel**.**fit(X\_train, y\_train)

***#Predict the value for new, unseen data***

knn\_pred **=** knnmodel**.**predict(X\_test)

***# Find Accuracy using accuracy\_score method***

knn\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, knn\_pred) **\*** 100, 2)

***#Fit the Decision Tree Classification Model***

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

dtmodel **=** DecisionTreeClassifier(criterion **=** "gini", random\_state **=** 50)

dtmodel**.**fit(X\_train, y\_train)

***#Predict the value for new, unseen data***

dt\_pred **=** dtmodel**.**predict(X\_test)

***# Find Accuracy using accuracy\_score method***

dt\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, dt\_pred) **\*** 100, 2)

***#Fit the Random Forest Classification Model***

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

rfmodel **=** RandomForestClassifier(n\_estimators **=** 100, criterion **=** 'entropy', random\_state **=** 0)

rfmodel**.**fit(X\_train, y\_train)

***#Predict the value for new, unseen data***

rf\_pred **=** rfmodel**.**predict(X\_test)

***# Find Accuracy using accuracy\_score method***

rf\_accuracy **=** round(metrics**.**accuracy\_score(y\_test, rf\_pred) **\*** 100, 2)

***# Compare Several models according to their Accuracies***

Model\_Comparison **=** pd**.**DataFrame({

'Model': ['Logistic Regression', 'Support Vector Machine', 'K-Nearest Neighbor',

'Decision Tree', 'Random Forest'],

'Score': [logmodel\_accuracy, svc\_accuracy, knn\_accuracy,

dt\_accuracy, rf\_accuracy]})

Model\_Comparison\_df **=** Model\_Comparison**.**sort\_values(by**=**'Score', ascending**=False**)

Model\_Comparison\_df **=** Model\_Comparison\_df**.**set\_index('Score')

Model\_Comparison\_df**.**reset\_index()

***# Predict the probability of Churn of each customer***

churn\_dataset['Probability\_of\_Churn'] **=** logmodel**.**predict\_proba(churn\_dataset[X\_test**.**columns])[:,1]

***# Create a Dataframe showcasing probability of Churn of each customer***

churn\_dataset[['customerID','Probability\_of\_Churn']]**.**head()

**NOTE :** In this phase of development,We have *Visualize Churn Rate by Internet Services, Visualize Churn Rate by Payment Method, Visualize Churn Rate by Contract Duration, Visualize Churn Rate by Contract Duration, Visualize Relation between Tenure & Churn rate, Perform One Hot Encoding using get dummies method, Perform Feature Scaling and One Hot Encoding, Machine Learning classification model libraries, Fit the logistic Regression Model, Fit the Support Vector Machine Model, Fit the Decision Tree Classification Model, Fit the Random Forest Classification Model, Predict the probability of Churn of each customer.*

**PHASE 4 SUBMISSION DONE BY :**

Jayapratha N

810021205036