PROJECT TITLE :CUSTOMER CHURN PREDICTION

PHASE 4: DEVELOPMENT PART 2

PROBLEM STATEMENT

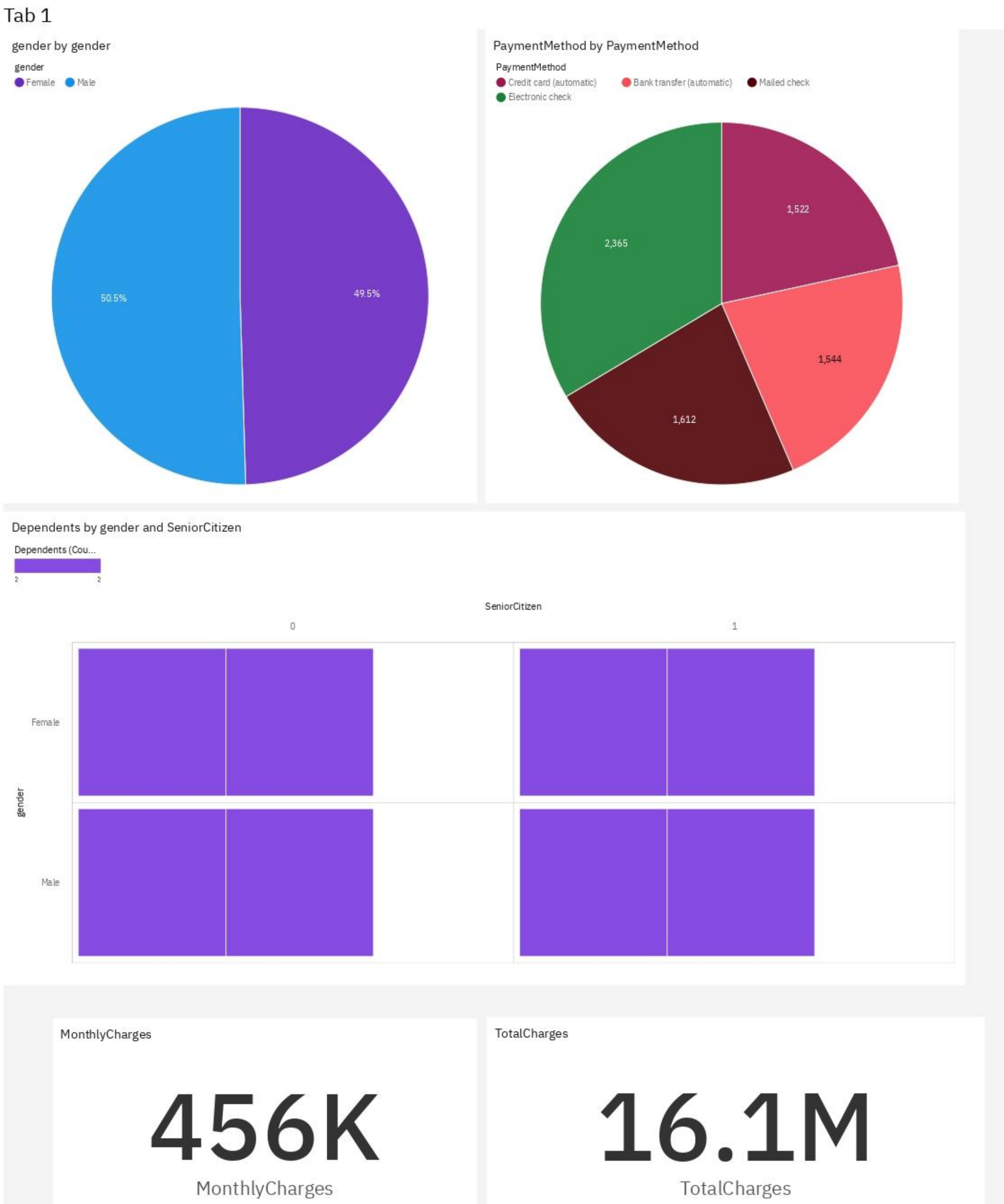
Phase 4: Development Part 2

In this part you will continue building your project.

- Continue building the analysis by creating visualizations using IBM Cognos and developing a predictive model.
- Create interactive dashboards and reports in IBM Cognos to visualize churn patterns, retention rates, and key factors influencing churn.
- Use machine learning algorithms to build a predictive model that identifies potential churners based on historical data and relevant features

.

WE HAVE CREATED DASHBOARD USING IBM COGNOS

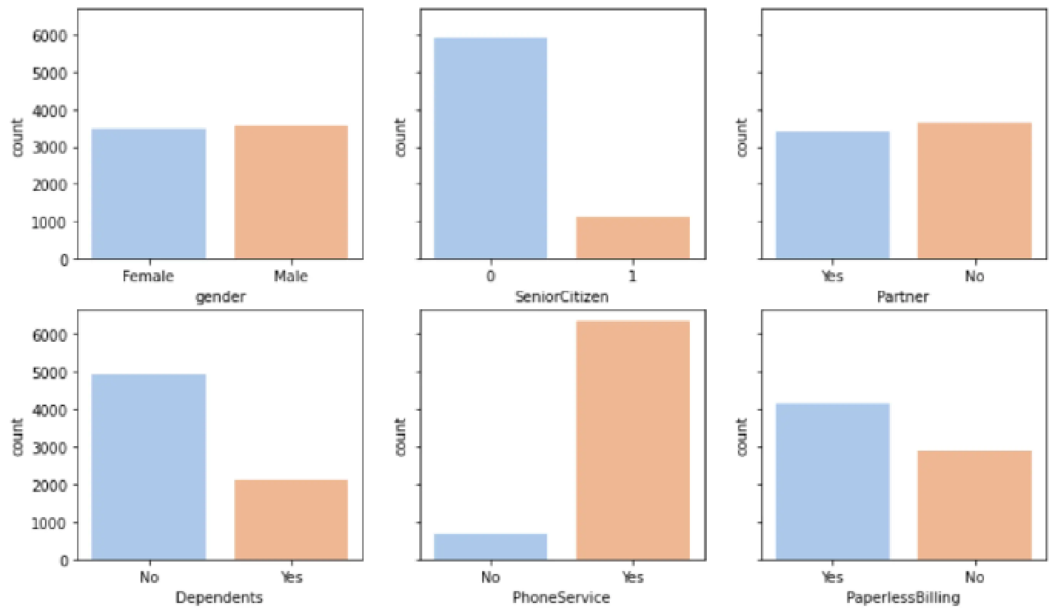


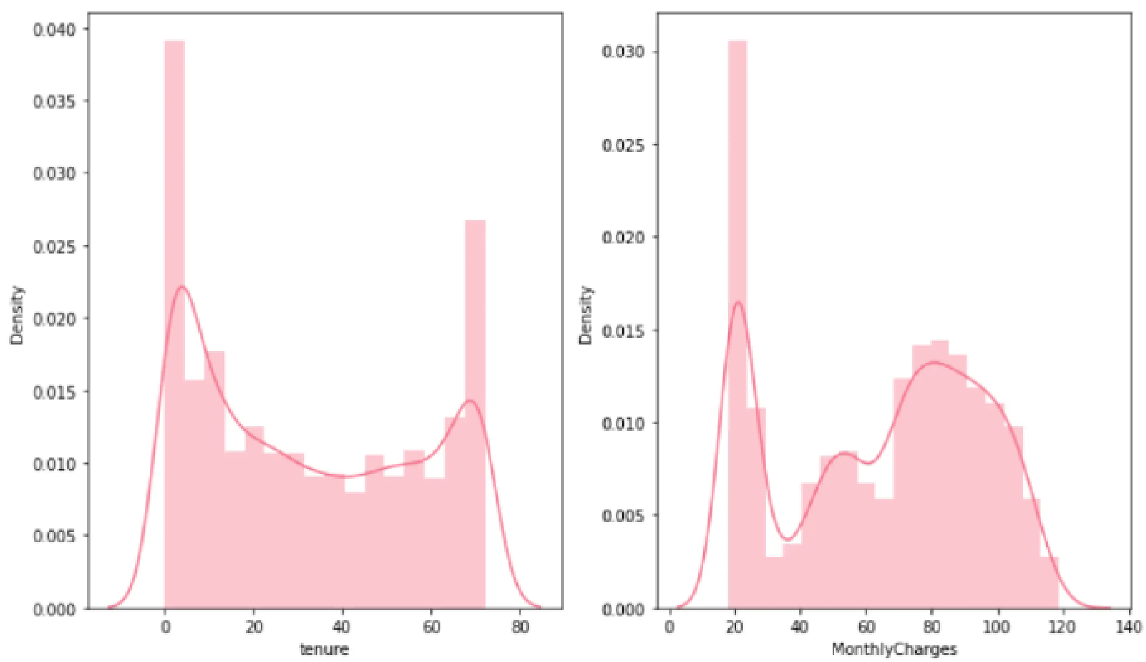
We have used SVM algorithm to build predictive modeling

```
# Loading Data

# Importing Dataset
data =
pd.read_csv("/kaggle/input/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.c
sv")
# Printing Data
data.head()
```

DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

```
with sns.color_palette("pastel"):
    fig, axes = plt.subplots(2, 3, figsize=(12, 7), sharey=True)
    sns.countplot("gender", data=data, ax=axes[0,0])
    sns.countplot("SeniorCitizen", data=data, ax=axes[0,1])
    sns.countplot("Partner", data=data, ax=axes[0,2])
    sns.countplot("Dependents", data=data, ax=axes[1,0])
    sns.countplot("PhoneService", data=data, ax=axes[1,1])
    sns.countplot("PaperlessBilling", data=data, ax=axes[1,2])


with sns.color_palette("husl"):
    fig, axes = plt.subplots(1,2, figsize=(12, 7))
    sns.distplot(data["tenure"], ax=axes[0])
    sns.distplot(data["MonthlyCharges"], ax=axes[1])
```





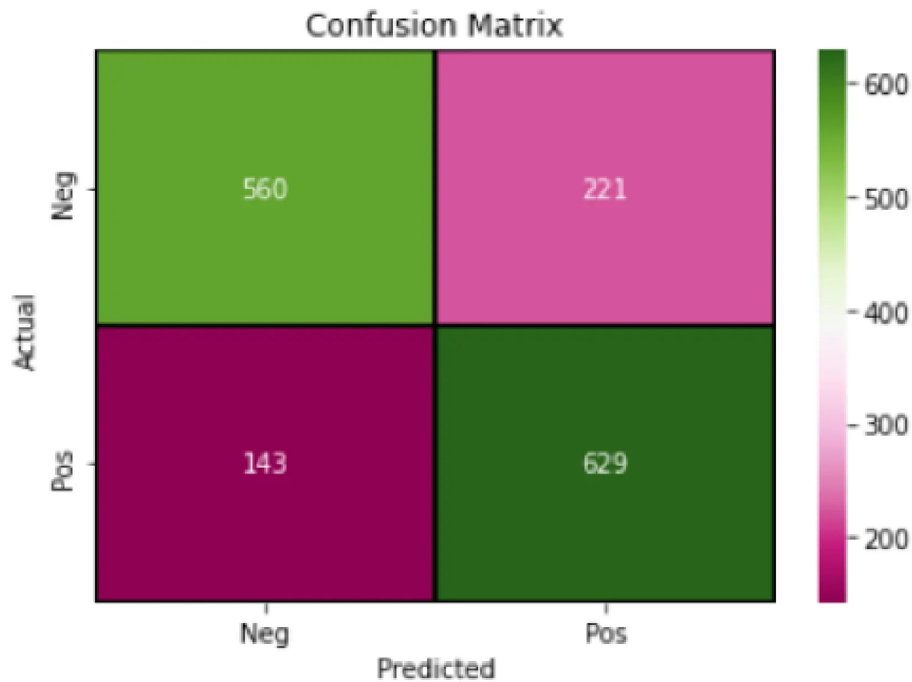
SVM CLASSIFIER

1. **SVM - SVM or Support Vector Machine is a supervised machine learning technique used for classification and regression. Finding a hyperplane in**

an N-dimensional space that classifies the data points is the goal of the SVM method. The number of features determines the hyperplane's size.

```
# Training the model using the optimal parameters discovered with SVM Classifier
svmclf =  SVC(C=3,class_weight='balanced', random_state=43)
svmclf.fit(X_train,y_train)

result2 = ["2.","SVM","Balanced using class weights"]
y_pred_tr = svmclf.predict(X_train)
print('Train accuracy SVM: ',accuracy_score(y_train,y_pred_tr))
result2.append(round(accuracy_score(y_train,y_pred_tr),2))

y_pred_test = svmclf.predict(X_test)
print('Test accuracy SVM: ',accuracy_score(y_test,y_pred_test))
result2.append(round(accuracy_score(y_test,y_pred_test),2))

recall = recall_score(y_test,y_pred_test)
print("Recall Score: ",recall)
result2.append(round(recall,2))

# Building a confusion matrix
matrix = confusion_matrix(y_test,y_pred_test)
ax=plt.subplot();
sns.heatmap(matrix, annot=True, fmt='d', linewidths=2, linecolor='black',
cmap='YlGnBu',ax=ax)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
ax.set_ylim(2.0,0)
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['Neg','Pos'])
ax.yaxis.set_ticklabels(['Neg','Pos'])
plt.show()
```

```
OUTPUT
Train accuracy SVM:  0.8186469584991473
Test accuracy SVM:  0.7656149388280747
Recall Score:  0.8147668393782384
```



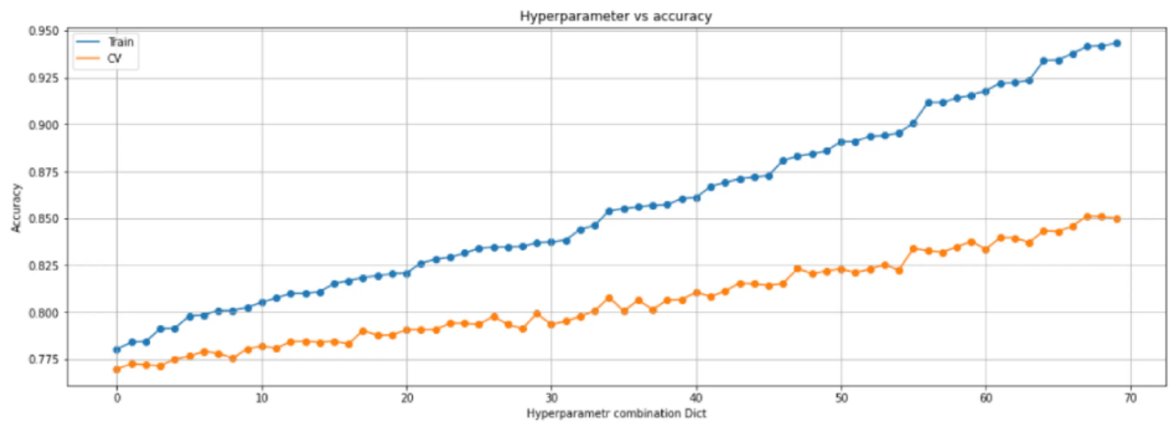
1. XG Boost - Formally speaking, XGBoost may be described as a decision tree-based ensemble learning framework that uses Gradient Descent as the underlying objective function. It offers excellent flexibility and efficiently uses computation to produce the mandated results.

```
# Grid Search To Get Best Hyperparameters

parameters = {"learning_rate"    : [0.10,0.20,0.30 ],\

            "max_depth"        : [ 3,5,10,20],\

            "n_estimators" : [ 100, 200, 300, 500],\

            "colsample_bytree" : [ 0.3, 0.5, 0.7 ] }

clf_xgb = XGBClassifier(scale_pos_weight=scale, eval_metric ='mlogloss')

grid = GridSearchCV(estimator=clf_xgb, param_grid=parameters,
scoring='accuracy',return_train_score=True,verbose=1)

grid.fit(X_train,y_train)


# plotting only the first 70 train scores
```

```
cv_result =
pd.DataFrame(grid.cv_results_).sort_values(by='mean_train_score',ascending=True)[:70]

param_list = list(cv_result['params'])

param_index = np.arange(70)

plt.figure(figsize=(18,6))

plt.scatter(param_index,cv_result['mean_train_score'])

plt.plot(param_index,cv_result['mean_train_score'],label='Train')

plt.scatter(param_index,cv_result['mean_test_score'])

plt.plot(param_index,cv_result['mean_test_score'],label="CV")

plt.title('Hyperparameter vs accuracy')

plt.grid()

plt.legend()

plt.xlabel('Hyperparametr combination Dict')

plt.ylabel('Accuracy')

plt.show()
```

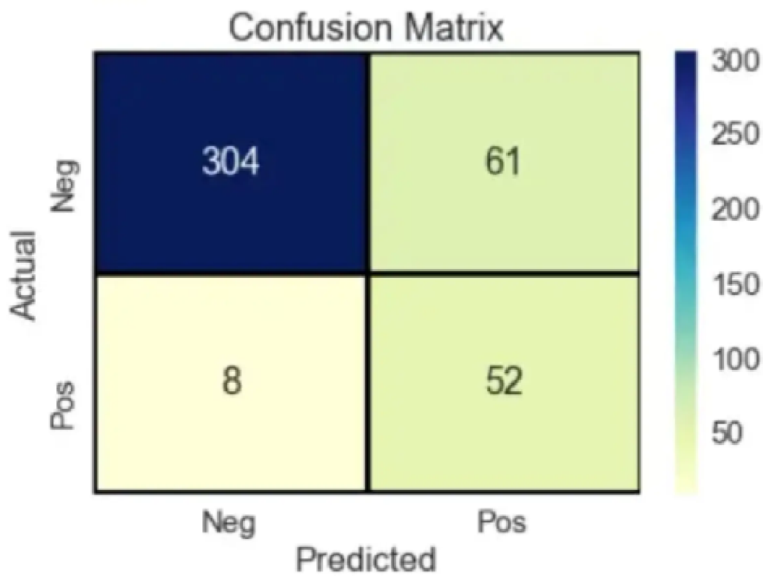
OUTPUT

Fitting 5 folds for each of 144 candidates, totaling 720 fits




```
# Using XG Boost

clf_xgb = XGBClassifier(learning_rate= best_parameters['learning_rate']
,max_depth=best_parameters ['max_depth'],
n_estimators=best_parameters['n_estimators'],
colsample_bytree=best_parameters['colsample_bytree'],
eval_metric='mlogloss',scale_pos_weight=scale)

clf_xgb.fit(X_train,y_train)


xgbresult = ["4.","XGBClassifier","Balanced using scale_pos_weight"]

y_pred_tr = clf_xgb.predict(X_train)

print('Train accuracy XGB: ',accuracy_score(y_train,y_pred_tr))

xgbresult.append(round(accuracy_score(y_train,y_pred_tr),2))


y_pred_test = clf_xgb.predict(X_train)

print('Test accuracy XGB: ',accuracy_score(y_test,y_pred_test))

xgbresult.append(round(accuracy_score(y_test,y_pred_test),2))


recall = recall_score(y_test,y_pred_test)

print("Recall Score: ",recall)
```

```
xgbresult.append(round(recall,2))


# Building confusion matrix

cm = confusion_matrix(y_test,y_pred_test)

ax=plt.subplot();

sns.heatmap(cm, annot=True, fmt='d', linewidths=2, linecolor='black',
cmap='YlGnBu',ax=ax)

ax.set_xlabel('Predicted')

ax.set_ylabel('Actual')

ax.set_ylim(2.0,0)

ax.set_title('Confusion Matrix')

ax.xaxis.set_ticklabels(['Neg','Pos'])

ax.yaxis.set_ticklabels(['Neg','Pos'])

plt.show()


Train accuracy XGB:  0.8543490619670268

Test accuracy: 0.80

Recall Score: 0.75
```



CONCLUSION

IN THIS PHASE WE HAVE CREATED DASHBOARD USING IBM COGNOS

AND WE USED MACHINE LEARNING ALGORITHM TO BUILD PREDICTIVE MODELING FOR CUSTOMER DATA AND WE USED SVM AND XG BOOST