CUSTOMER CHURN PREDICTION

A PROJECT REPORT

PHASE V

A report submitted in fulfilment of the project

Of

DATA ANALYTICS WITH COGNOS - GROUP 1

In

NAAN MUDHALVAN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY

CHENNAI 600025

submission on

30-OCT-2023

Submitted by

SRIVATHSAN G	2021103583
SUNIL KUMAR S	2021103586
UMA T	2021103593
VALARMATHI B	2021103594
YOGA VARSHA M	2021103601

ABSTRACT:

Customer churn prediction is a crucial task for businesses seeking to retain and expand their customer base. This predictive modeling approach leverages historical customer data, such as transaction history and demographics, to forecast the likelihood of a customer discontinuing their engagement with a company. By identifying potential churners in advance, businesses can proactively implement targeted retention strategies, ultimately reducing customer attrition and optimizing long-term profitability. This abstract highlights the importance of customer churn prediction as a strategic tool for enhancing customer retention and maximizing overall business success.

INTRODUCTION:

In the fast-paced world of business, customer retention is of paramount importance. Customer churn, or the loss of customers, can be detrimental to a company's bottom line and long-term success. To address this challenge, businesses are turning to customer churn prediction, a data-driven approach that uses historical customer information and advanced analytics to forecast which customers are most likely to leave.

This project focuses on developing predictive models and strategies using machine learning and artificial intelligence to identify potential churners, enabling businesses to take proactive measures to retain valuable customers. In the following sections, we will explore the methods and tools for effective customer churn prediction, providing a roadmap for businesses aiming to enhance customer retention.

OBJECTIVE:

The objective of customer churn prediction is to leverage historical customer data and advanced analytics to identify customers at risk of discontinuing their relationship with a company. By proactively pinpointing potential churners, businesses aim to implement targeted strategies that reduce customer attrition, enhance customer retention, and ultimately optimize long-term profitability.

DESIGN THINKING PROCESS:

Design thinking is a structured approach to problem-solving and innovation. In this project, we follow the design thinking process:

1. **Empathize:**

   Begin by understanding the customer's perspective. Gather data on customer behaviors, feedback, and pain points. Conduct surveys, interviews, and analyze customer interactions to gain insights into why customers churn. This empathetic phase is crucial for framing the problem from the customer's point of view.

2. **Define:**

   Based on the insights gathered, define the problem. Clearly articulate the goals of the churn prediction project, such as reducing churn rates, increasing customer satisfaction, or retaining high-value customers.

3. **Ideate:**

   Encourage brainstorming and generate a wide range of ideas and hypotheses to address the churn problem. Consider various data sources, predictive modeling techniques, and potential intervention strategies. Cross-disciplinary teams can help bring diverse perspectives to the ideation phase.

4. **Prototype:**

   Develop preliminary churn prediction models based on the chosen data sources and techniques. These models need not be perfect but serve as early iterations to test the feasibility of your ideas. They should allow for rapid experimentation and learning.

5. **Test:**

   Implement your prototype models and evaluate their performance in a real-world context. Assess how well they predict customer churn and whether the strategies to retain customers are effective. This phase involves iterative testing and feedback to refine your approach.

6. **Iterate:**

   Based on the test results, make necessary improvements to the prediction models and strategies. Iterate on the prototypes, gather more data, and adjust your approach as you gain a deeper understanding of what works best.

7. **Implement:**

   Once you have a refined churn prediction model and retention strategies, implement them in your business operations. Ensure that relevant departments are aligned with the new approach and understand how to act upon the churn predictions effectively.

8. **Monitor and Learn:**

   Continuously monitor the performance of your churn prediction model and the outcomes of your retention strategies. Learn from the data and feedback, and be prepared to make adjustments as the business and customer landscape evolves.

9. **Scale:**

   If the churn prediction model and retention strategies prove successful, scale your approach to cover a broader customer base. This may involve integrating it into various customer touchpoints and automating decision-making processes.

10. **Feedback Loop:**

    Maintain a feedback loop with customers and internal stakeholders to gather insights on the impact of churn prediction and retention efforts. Use this feedback to drive further improvements and innovation.

IMPLEMENTATION :

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler


from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier


from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score,

from sklearn.ensemble import VotingClassifier
```

In [2]:

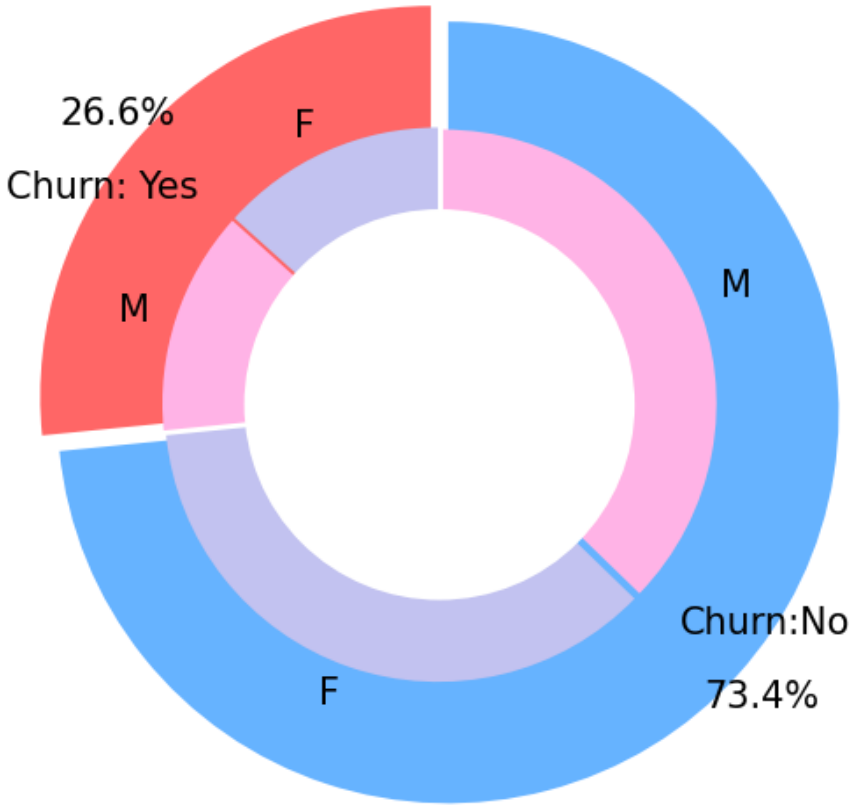
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import f1_score, precision_score, recall_score, fbeta_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import KFold
from sklearn import feature_selection
from sklearn import model_selection
from sklearn import metrics
from sklearn.metrics import classification_report, precision_recall_curve
from sklearn.metrics import auc, roc_auc_score, roc_curve
from sklearn.metrics import make_scorer, recall_score, log_loss
from sklearn.metrics import average_precision_score
#Standard libraries for data visualization:
```

In [6]:
```
data = pd.read_csv("/Users/sunilkumars/Downloads/WA_Fn-UseC_-Telco-Customer-Churn.csv")
data.head()
```

Out[6]:

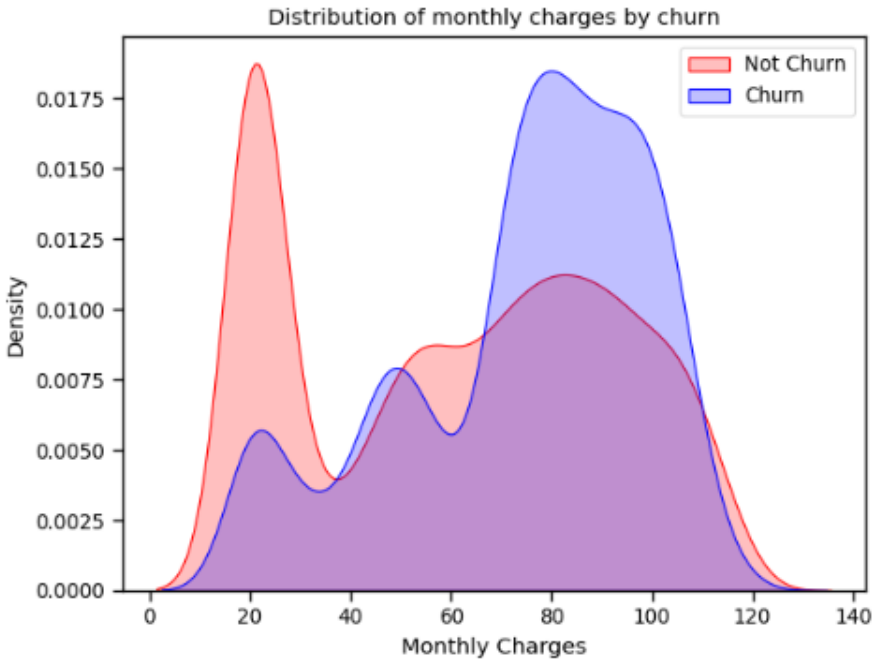
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...

5 rows × 21 columns



In [42]:

```
sns.set_context("paper",font_scale=1.1)
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'No') ],
                 color="Red", shade = True);
ax = sns.kdeplot(data.MonthlyCharges[(data["Churn"] == 'Yes') ],
                 ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Monthly Charges');
ax.set_title('Distribution of monthly charges by churn');
```

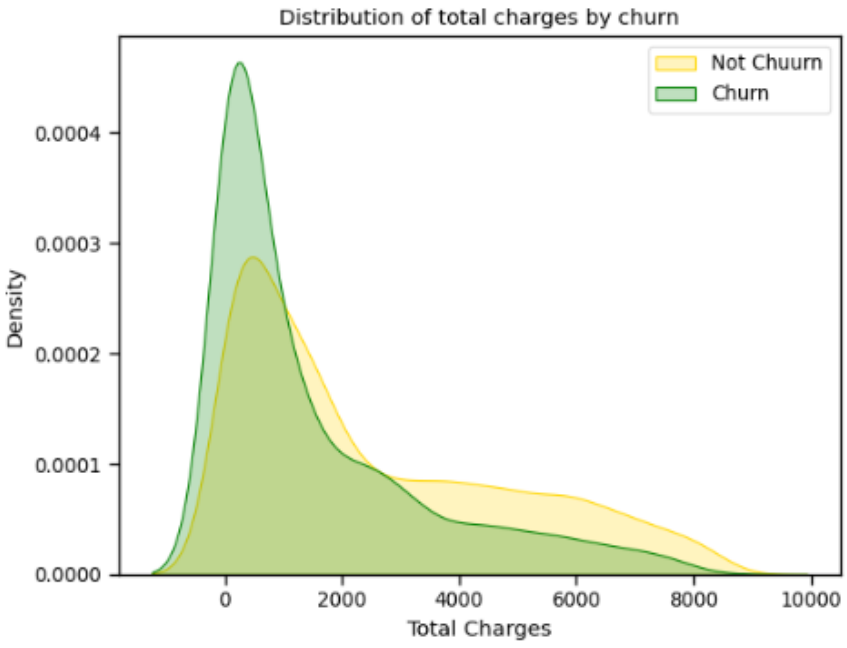


In [43]:

```
#Customers with higher monthly charges are more likely to churn
```

In [44]:

```
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'No') ],
                 color="Gold", shade = True);
ax = sns.kdeplot(data.TotalCharges[(data["Churn"] == 'Yes') ],
                 ax =ax, color="Green", shade= True);
ax.legend(["Not Chuurn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Total Charges');
ax.set_title('Distribution of total charges by churn');
```



In [45]:

```
fig = px.box(data, x='Churn', y = 'tenure')

# Update yaxis properties
fig.update_yaxes(title_text='Tenure (Months)', row=1, col=1)
# Update xaxis properties
fig.update_xaxes(title_text='Churn', row=1, col=1)

# Update size and title
fig.update_layout(autosize=True, width=750, height=600,
    title_font=dict(size=25, family='Courier'),
    title='<b>Tenure vs Churn</b>',
)

fig.show()
```

In [46]:

```
#New customers are more likely to churn
```

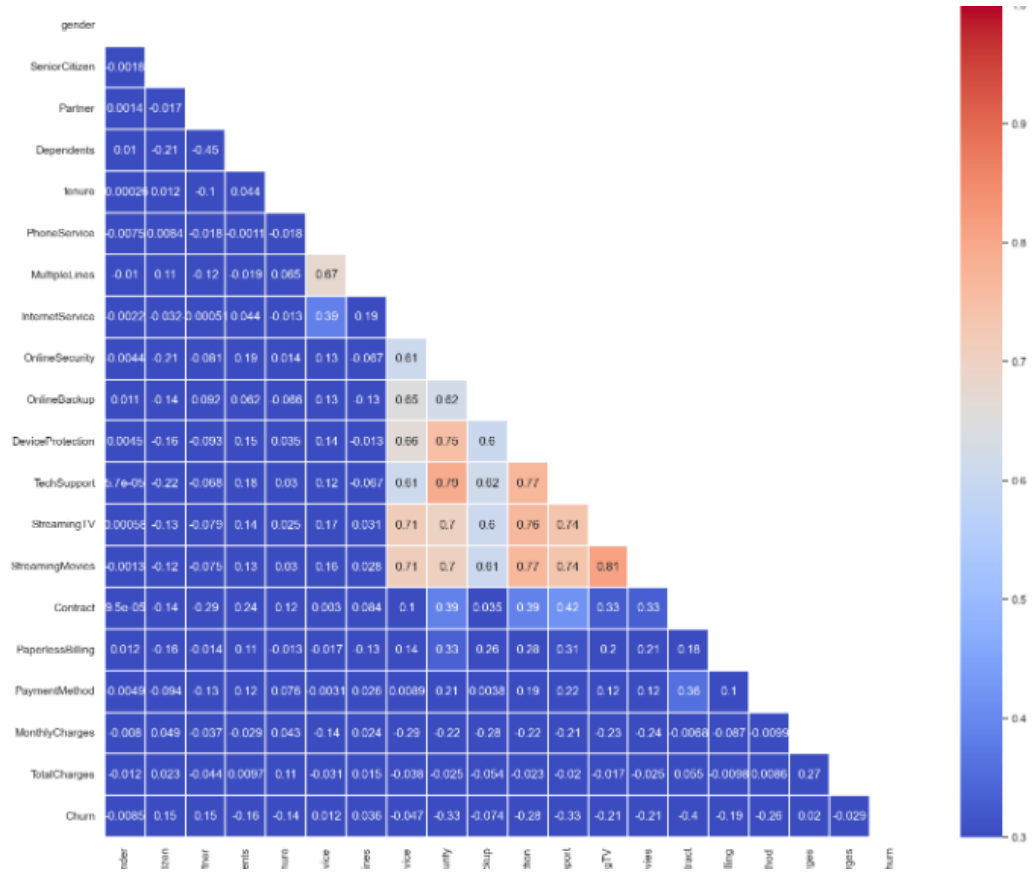
In [47]:

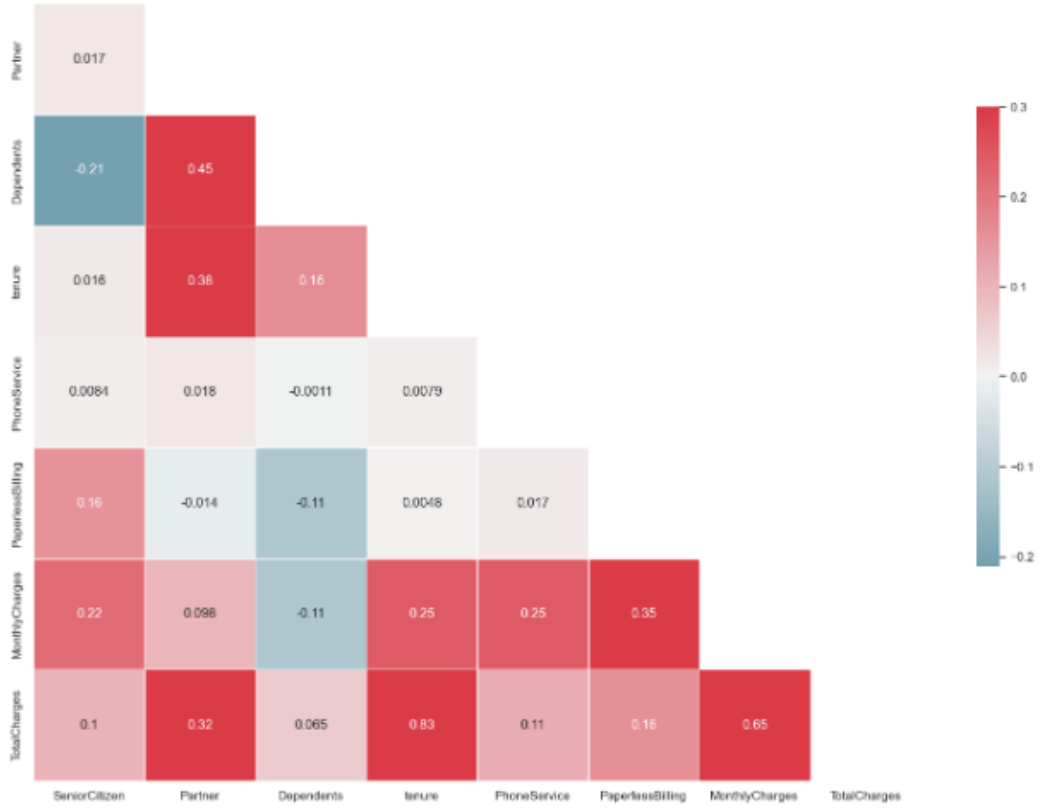
```
#Create a label encoder object
le = LabelEncoder()
# Label Encoding will be used for columns with 2 or less unique
values
le_count = 0
for col in data.columns[1:]:
    if data[col].dtype == 'object':
        if len(list(data[col].unique())) <= 2:
            le.fit(data[col])
            data[col] = le.transform(data[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
```

```
6 columns were label encoded.
```

In [49]:

```
#Set and compute the Correlation Matrix:
sns.set(style="white")
plt.figure(figsize=(18, 15))

corr = data.apply(lambda x: pd.factorize(x)[0]).corr()

mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, linewidths=.2, cmap='cc
```



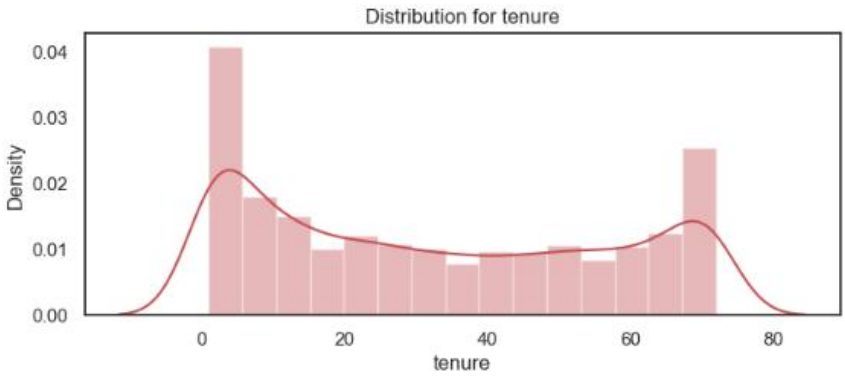


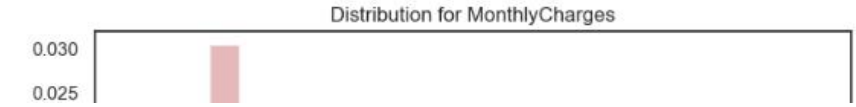
In [55]:
```
X = data.drop(columns = "Churn")
y = data["Churn"].values
```

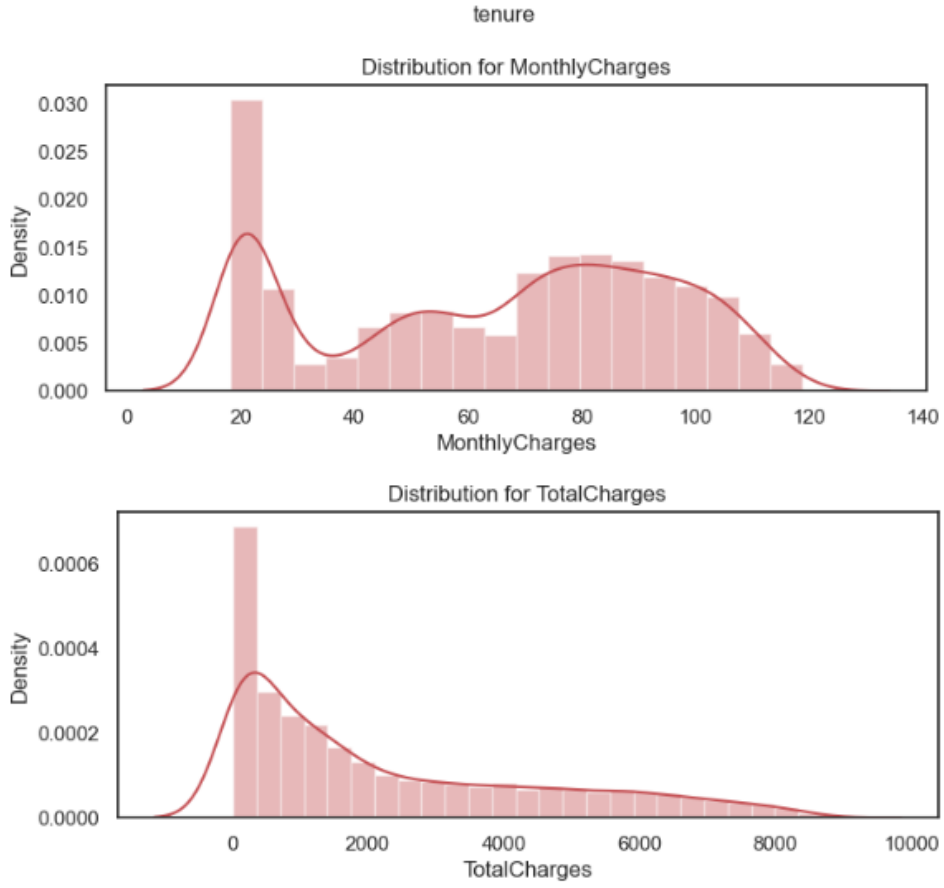
In [56]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 4, stratify =y)
```

In [57]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

In [58]:
```
col =  ["tenure", 'MonthlyCharges', 'TotalCharges']
for features in col :distplot(features, data)
```





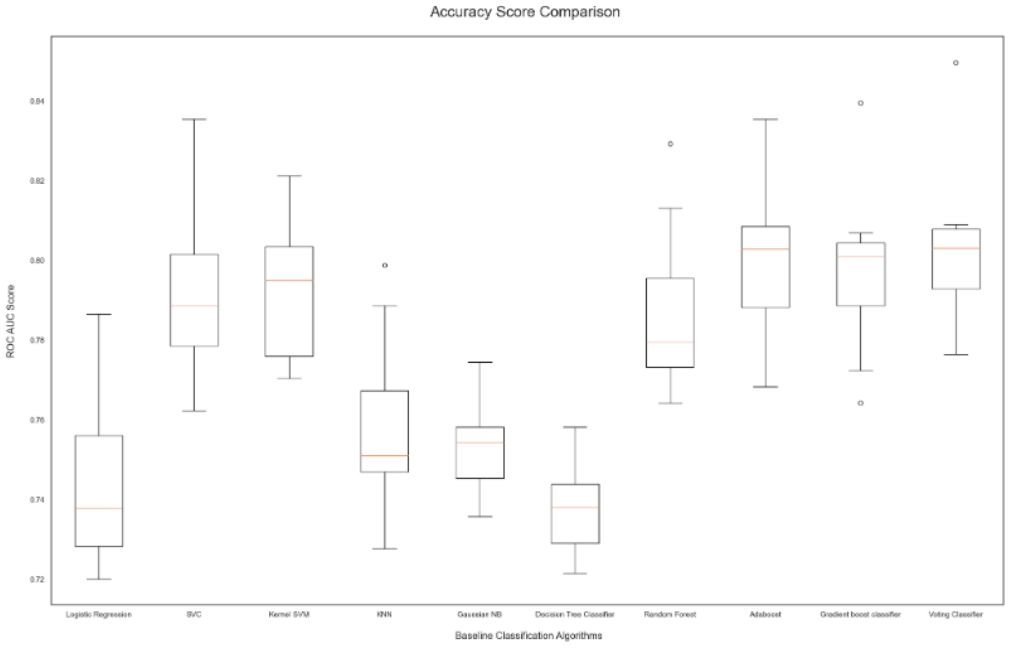


In [67]:
```
#Evaluating the model Results
```

In [69]:
```
acc_results =[]
auc_results =[]
names = []

result_col = ["Algorithm", "ROC AUC Mean", "ROC AUC STD", "Accuracy Mean", "Accuracy STD"]
model_results = pd.DataFrame(columns = result_col)

i=0
# K- fold cross validation

for name, model in models:
    names.append(name)
    kfold = model_selection.KFold(n_splits=10, shuffle=True, random_state=0)
    kfold = model_selection.KFold(n_splits=10)
    cv_acc_results = model_selection.cross_val_score(model, X_train, y_train,
                     cv = kfold, scoring="accuracy")
    cv_auc_results = model_selection.cross_val_score(model, X_train, y_train,
                     cv = kfold, scoring="roc_auc")
    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)

    model_results.loc[i] = [name,
                            round(cv_auc_results.mean()*100,2),
                            round(cv_auc_results.std()*100,2),
                            round(cv_acc_results.mean()*100,2),
                            round(cv_acc_results.std()*100,2)]
    i+=1

model_results.sort_values(by = ['ROC AUC Mean'], ascending=False)
```

Out[69]:

	Algorithm	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD
9	Voting Classifier	84.93	1.38	80.23	1.89
8	Gradient boost classifier	84.71	1.41	79.74	1.96
7	Adaboost	84.55	1.25	80.09	1.77
0	Logistic Regression	84.39	1.47	74.38	1.94
1	SVC	82.99	2.07	79.11	2.01
6	Random Forest	82.75	2.01	78.67	1.98
4	Gaussian NB	82.32	1.28	75.38	1.23
2	Kernel SVM	79.65	2.12	79.26	1.67
3	KNN	77.13	1.49	75.86	2.03
5	Decision Tree Classifier	66.67	1.07	73.73	1.12



In [80]:

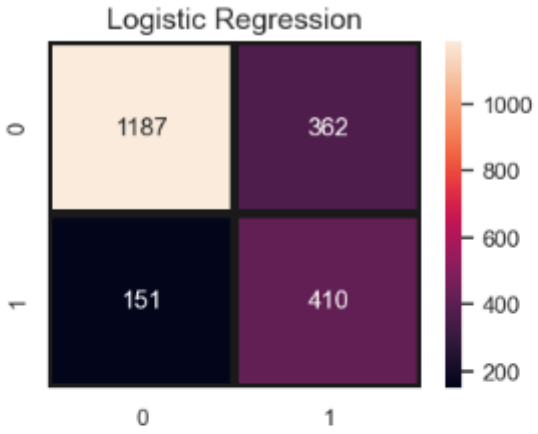
```
#evaluation of results
def model_evaluation(y_test, y_pred, model_name):
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    f2 = fbeta_score(y_test, y_pred, beta = 2.0)

    results = pd.DataFrame([[model_name, acc, prec, rec, f1, f2]],
                      columns = ["Model", "Accuracy", "Precision", "Recall",
                                 "F1 SCore", "F2 Score"])
    results = results.sort_values(["Precision", "Recall", "F2 Score"], ascending = False)
    return results
```

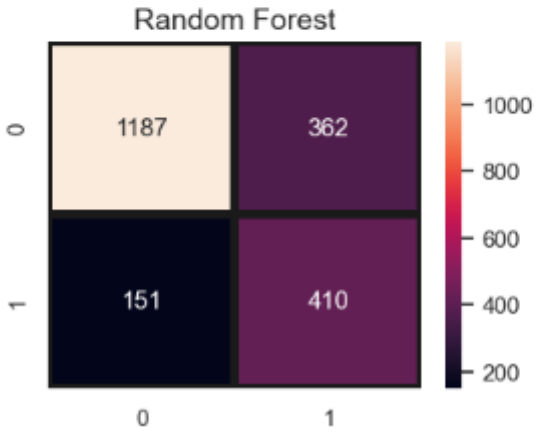
In [81]:

```
# Logistic regression
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

#SVC

classifier2 = SVC(kernel = 'linear', random_state = 0)
classifier2.fit(X_train, y_train)
y_pred2 = classifier2.predict(X_test)

#knn

classifier3 = KNeighborsClassifier(n_neighbors=22, metric="minkowski", p=2)
classifier3.fit(X_train, y_train)
y_pred3 = classifier3.predict(X_test)
```

In [82]:

```
lr = model_evaluation(y_test, y_pred, "Logistic Regression")
svm = model_evaluation(y_test, y_pred2, "SVM (Linear)")
knn = model_evaluation(y_test, y_pred3, "K-Nearest Neighbours")
k_svm = model_evaluation(y_test, y_pred4, "Kernel SVM")
nb = model_evaluation(y_test, y_pred5, "Naive Bayes")
dt = model_evaluation(y_test, y_pred6, "Decision Tree")
rf = model_evaluation(y_test, y_pred7, "Random Forest")
ab = model_evaluation(y_test, y_pred8, "Adaboost")
gb = model_evaluation(y_test, y_pred9, "Gradient Boost")
vc = model_evaluation(y_test, y_pred10, "Voting Classifier")
```





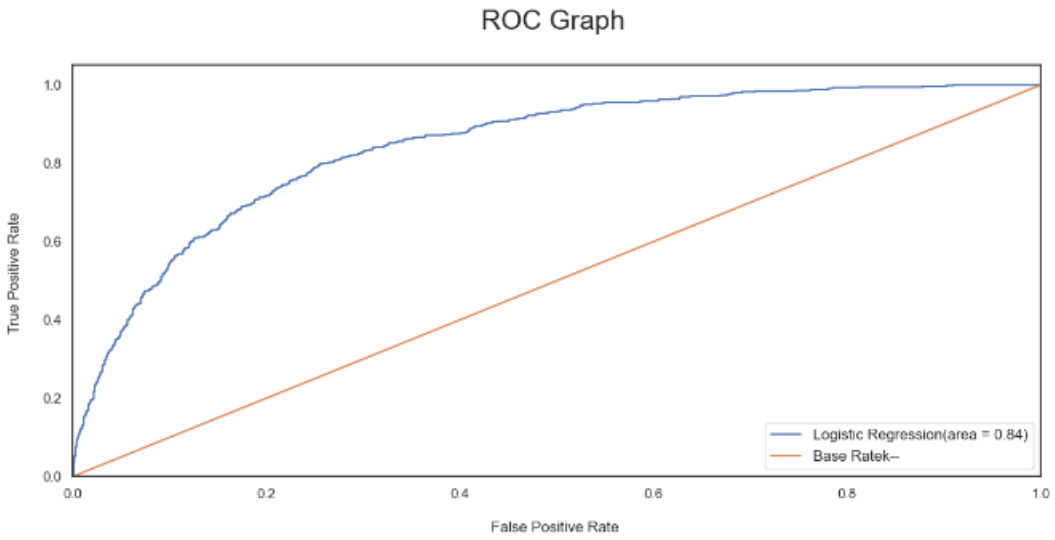
In [94]:
```
k_fold_cross_validation(classifier, "Logistic regression")
```

```
Logistic regression accuracy: 0.80 (+/- 0.04)
```

In [95]:
```
k_fold_cross_validation(classifier4, "Kernel SVM")
```

```
Kernel SVM accuracy: 0.80 (+/- 0.03)
```

In [97]:
```
preds = [y_pred, y_pred3,  y_pred5, y_pred6, y_pred7,
             y_pred8, y_pred9, y_pred10]
classifiers = [classifier , classifier3, classifier5, classifier6, classifier7,
            classifier8, classifier9, classifier10]
model_names_ = ["Logistic Regression",  "K-Nearest Neighbours","Naive Bayes",
              "Decision Tree", "Random Forest", "Adaboost", "Gradient Boost",  "Voting Classifier"]

for i, j, k in zip(classifiers, model_names_, predictions):
    ROC_curve(i, j, k)
```



In [98]:
```
# Cross validation

from sklearn.model_selection import cross_val_score

# Function that will track the mean value and the standard deviation of the accuracy
def cvDictGen(functions, scr, X_train = X, y_train = y, cv = 5):
    cvDict = {}
    for func in functions:
        cvScore = cross_val_score(func, X_train, y_train, cv = cv, scoring = scr)
        cvDict[str(func).split('(')[0]] = [cvScore.mean(), cvScore.std()]

    return cvDict
```

In [99]:
```
cvD = cvDictGen(classifiers, scr = 'roc_auc')
cvD
```

Out[99]:
```
{'LogisticRegression': [0.8415493547591085, 0.010311381652881136],
 'KNeighborsClassifier': [0.791184838669004, 0.008273675467323766],
 'GaussianNB': [0.8232386881685605, 0.00741678015498337],
 'DecisionTreeClassifier': [0.6470213137060805, 0.02196953973039052],
 'RandomForestClassifier': [0.8197874155380965, 0.011556155864106703],
 'AdaBoostClassifier': [0.8445838813774079, 0.01125665302188384],
 'GradientBoostingClassifier': [0.8447254381880706, 0.010738014069847618],
 'VotingClassifier': [0.8469030053182566, 0.010804068762983171]}
```

DEVELOPMENT PHASES:

1. Analysis Objectives:

- ✓ The analysis objectives in the development phases of customer churn prediction revolve around uncovering influential factors, evaluating model performance, and ensuring data quality.
- ✓ These objectives guide the selection of features and customer segments, assess model accuracy and interpretability, and address data-related issues, collectively enabling the creation of robust churn prediction models and effective retention strategies to minimize customer attrition and enhance customer relationships.

2. Data Collection Process:

- ✓ The data collection process in the development phases of customer churn prediction involves gathering a diverse set of customer data, such as transaction history, demographics, customer interactions, and relevant behavioral attributes.
- ✓ This data is sourced from various internal databases and systems, as well as external sources when necessary.
- ✓ The goal is to create a comprehensive dataset that provides a holistic view of customer interactions and behaviors, enabling accurate predictive modeling for identifying potential churners and developing tailored retention strategies.

3. Data Visualization using IBM Cognos:

- ✓ IBM Cognos is employed to create interactive and informative data visualizations.
- ✓ Dashboards and reports are designed to present customer churn trends in a clear and accessible manner.

4. Python Code Integration:

- ✓ Machine learning models are developed using Python, with popular libraries like Scikit-Learn.
- ✓ These models are trained to predict customer churn information.
- ✓ The Python code is integrated into the system to provide real-time analysis and predictions.

5. Improving User Experience:

- ✓ Enhancing user experience in the development phases of customer churn prediction involves creating intuitive interfaces, streamlined reporting, and fostering user engagement.
- ✓ User-friendly dashboards make data exploration and model evaluation more accessible, while automated reporting keeps stakeholders informed and minimizes manual efforts.
- ✓ Additionally, a feedback mechanism that encourages collaboration and input from users ensures that their insights and needs are integrated, ultimately optimizing the overall experience and effectiveness of the churn prediction process.

EXPLANATION:

- ❖ **Data Collection Process:**

The data collection process in customer churn prediction entails a comprehensive approach. It begins with identifying relevant data sources, which may include internal databases, transaction records, customer profiles, and external data providers. Data is then extracted and integrated, involving cleaning to address missing values, duplicates, and inconsistencies to ensure data quality. Subsequently, data transformation occurs to prepare the dataset for analysis, which can involve encoding categorical variables, creating derived features, and structuring the data for predictive modeling. This process aims to assemble a cohesive and high-quality dataset that encompasses diverse customer information, enabling accurate churn prediction and the development of tailored retention strategies.

- ❖ **Data Visualization using IBM Cognos:**

IBM Cognos is a powerful tool for translating raw data into visually comprehensible information. Through the creation of interactive dashboards, reports, and data visualizations, customer churn trends become evident to stakeholders. These visualizations not only enhance understanding but also facilitate informed decision-making.

- ❖ **Python Code Integration for Analysis:**

Machine learning models, developed using Python and libraries like Scikit-Learn, play a pivotal role in this project. These models are trained using dataset allowing for real-time

analysis of customer churn .The Python code is seamlessly integrated into the system, ensuring that the analysis is conducted promptly and accurately.

- **Improving User Experience:**

Improving user experience in the context of customer churn prediction is pivotal for successful project implementation. This involves designing intuitive and visually informative dashboards and interfaces that empower stakeholders, including data analysts, data scientists, and business decision-makers, to easily access and interpret the results. Interactive tools and clear data visualizations simplify the often complex analytical tasks, enhancing user engagement and understanding. Implementing automated reporting mechanisms ensures timely and consistent communication of critical insights and project milestones, reducing the need for manual monitoring and enabling more informed decision-making. Moreover, creating a feedback loop that encourages continuous collaboration and input from end-users helps refine predictive models and strategies based on real-world user insights, making the entire churn prediction process more user-centric and effective in reducing customer attrition.

CONCLUSION:

In conclusion, the customer churn prediction project empowers businesses to safeguard their customer base, adapt to market changes, and secure long-term success through data-driven insights.