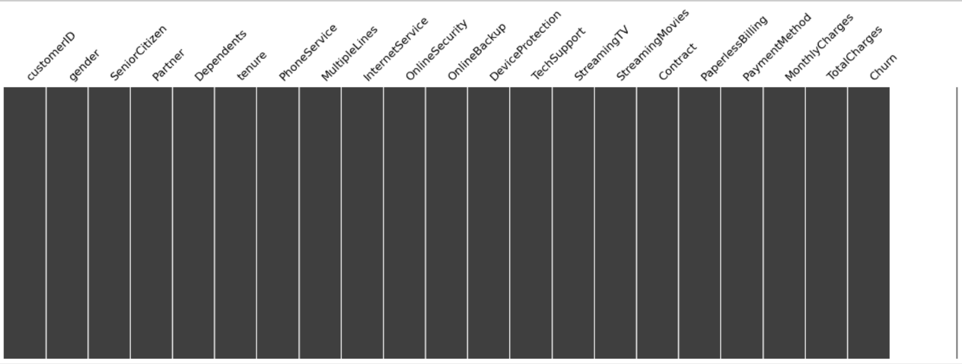
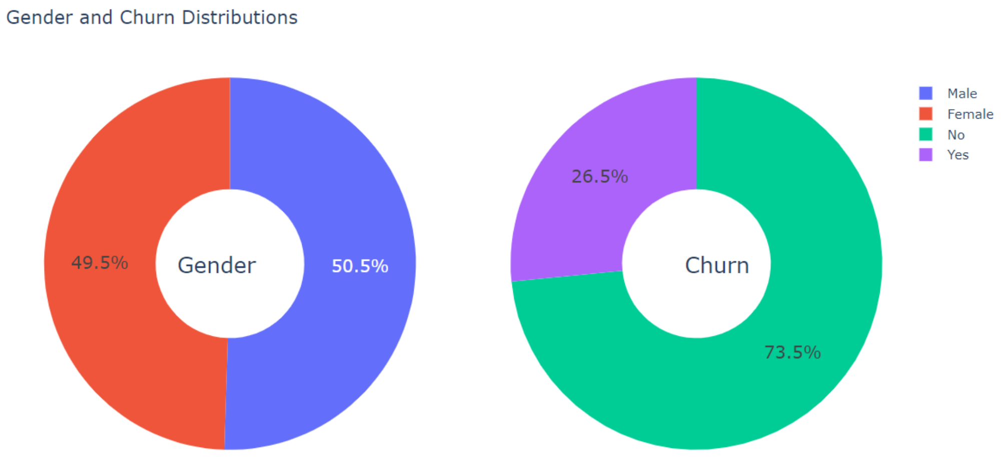
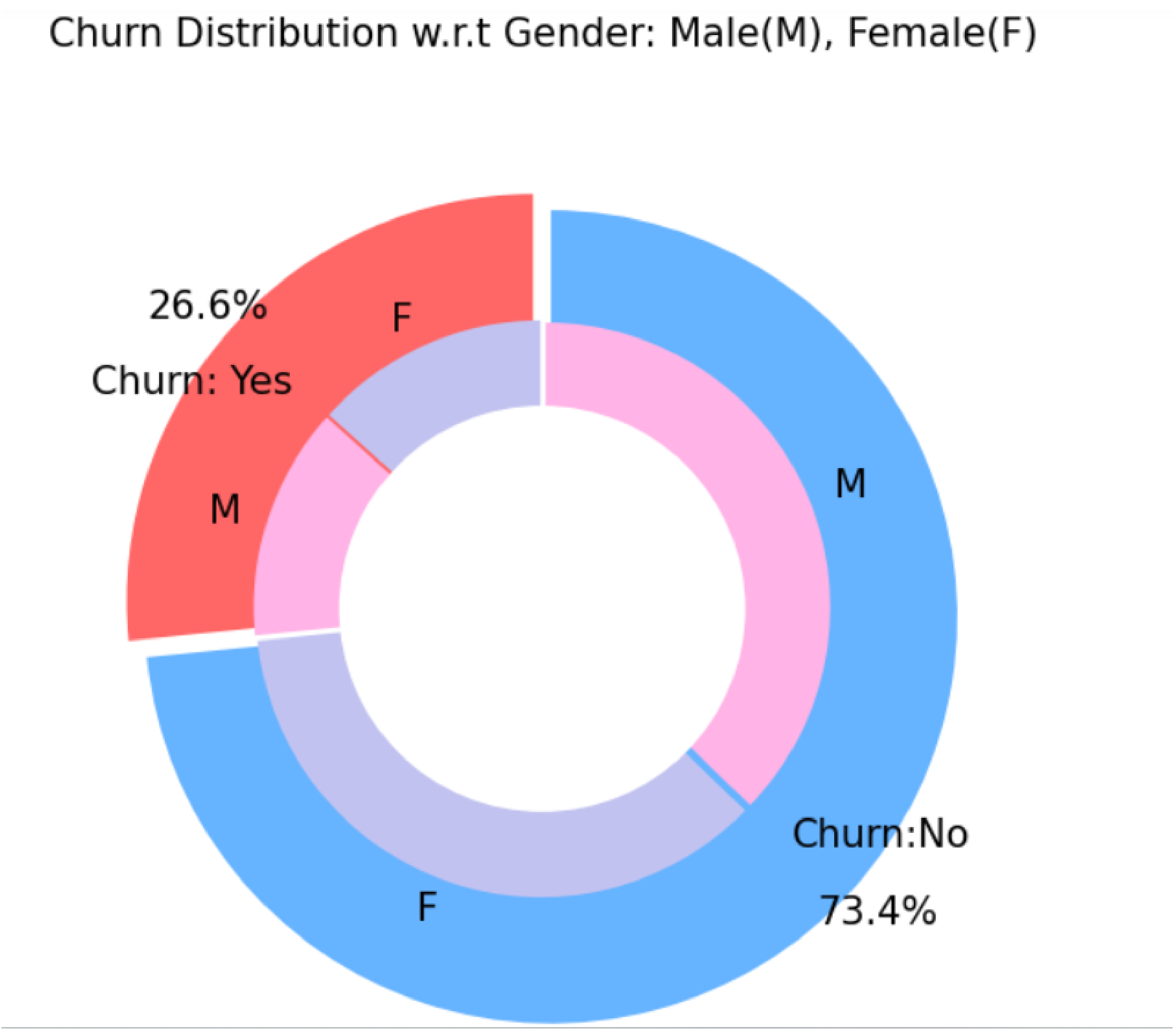
```
In [8]: msno.matrix(df);
```

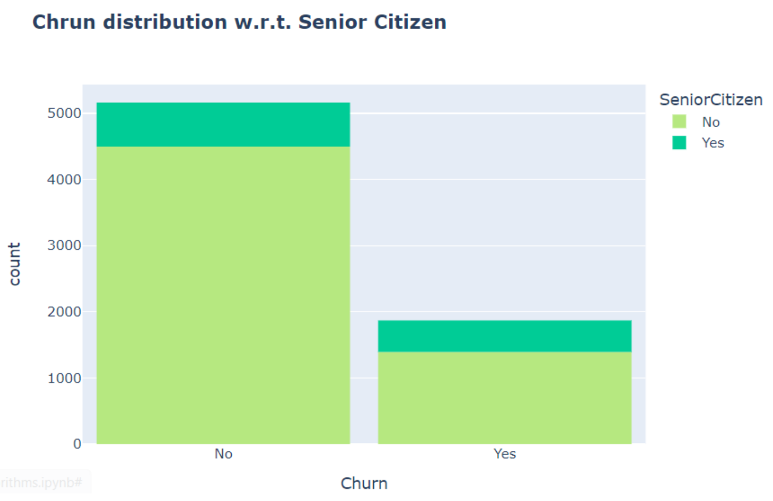


```
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='Gender', x=0.16, y=0.5, font_size=20, showarrow=False),
                 dict(text='Churn', x=0.84, y=0.5, font_size=20, showarrow=False)])
fig.show()
```

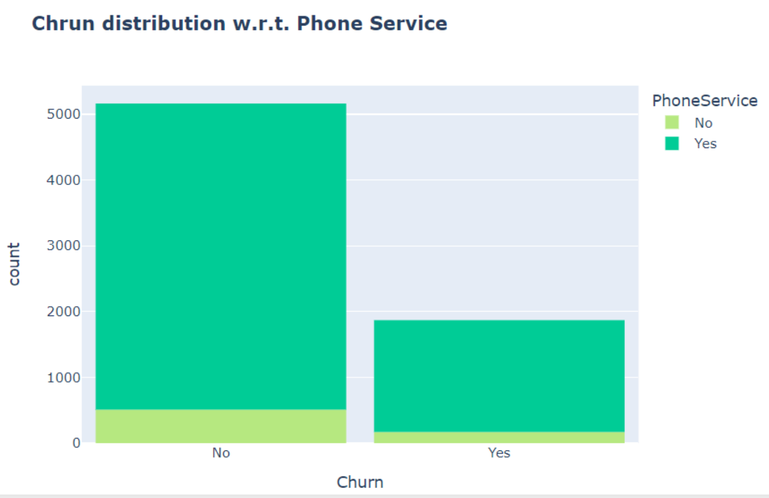




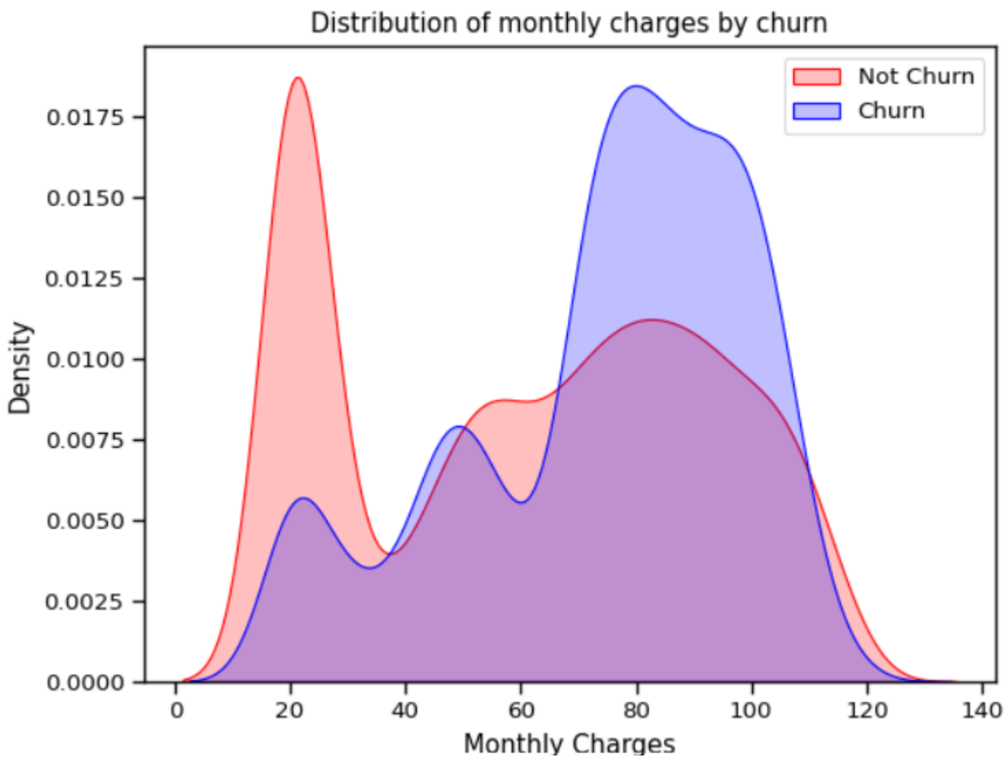
```
In [40]: color_map = {"Yes": '#00CC96', "No": '#B6E880'}
         fig = px.histogram(df, x="Churn", color="SeniorCitizen", title="<b>Chrun distribution w.r.t. Senior Citizen</b>", color_discrete_
         fig.update_layout(width=700, height=500, bargap=0.1)
         fig.show()
```



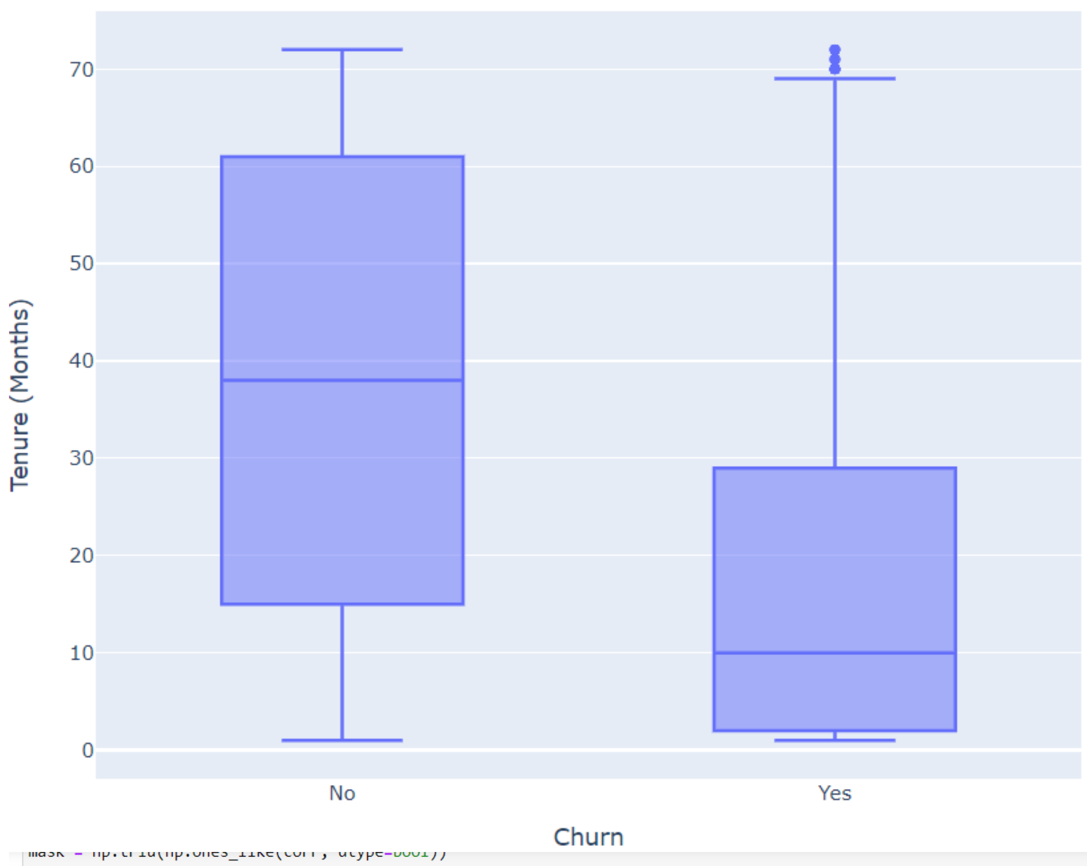
```
In [41]: color_map = {"Yes": '#00CC96', "No": '#B6E880'}
         fig = px.histogram(df, x="Churn", color="PhoneService", title="<b>Chrun distribution w.r.t. Phone Service</b>", color_discrete_ma
         fig.update_layout(width=700, height=500, bargap=0.1)
         fig.show()
```



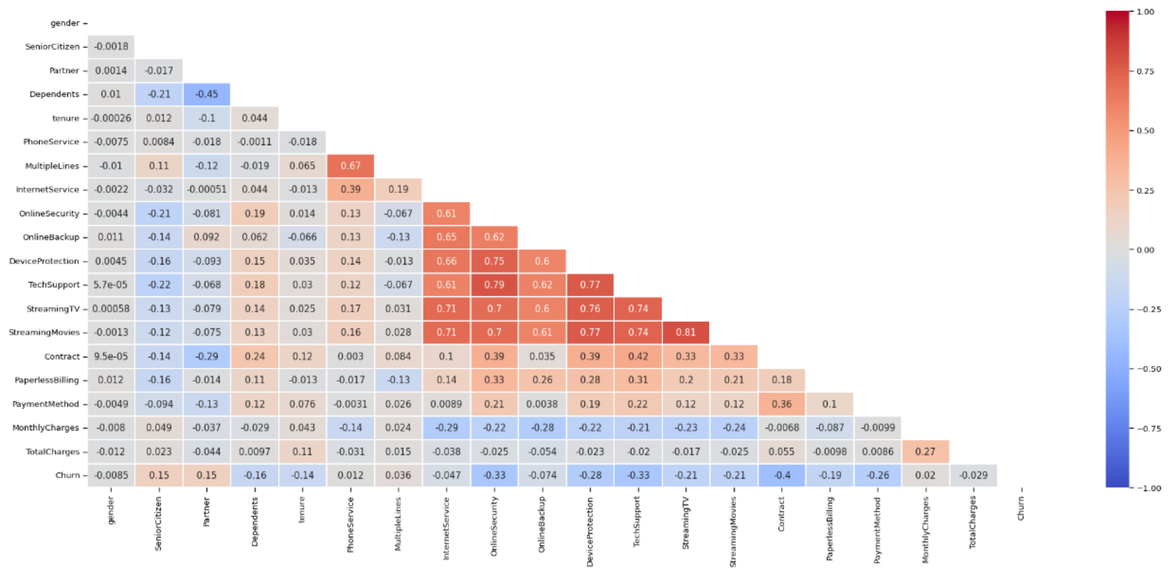
```
ax = sns.kdeplot(df.MonthlyCharges[(df["Churn"] == 'Yes') ],
                ax =ax, color="Blue", shade= True);
ax.legend(["Not Churn","Churn"],loc='upper right');
ax.set_ylabel('Density');
ax.set_xlabel('Monthly Charges');
ax.set_title('Distribution of monthly charges by churn');
```



Tenure vs Churn

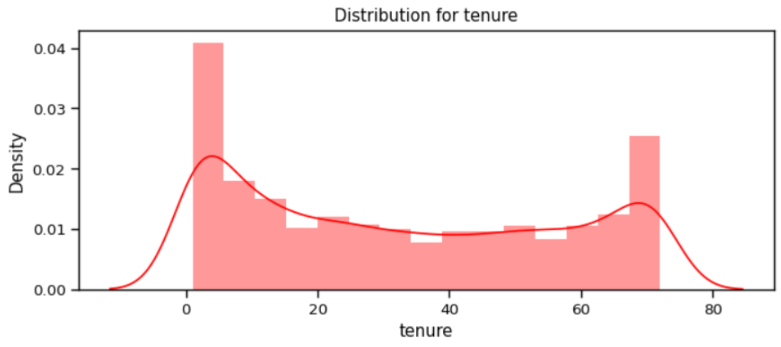


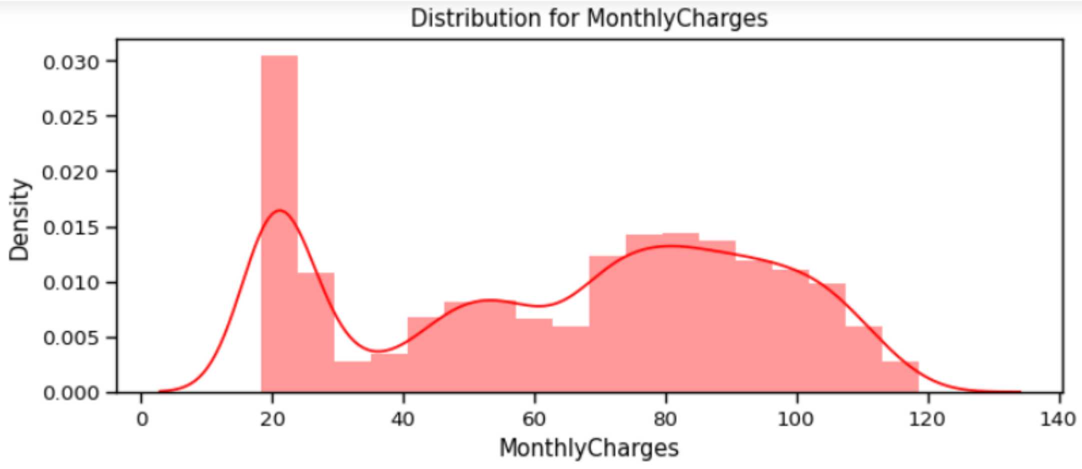
```
mask = np.triu(np.ones_like(corr, dtype=bool))

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

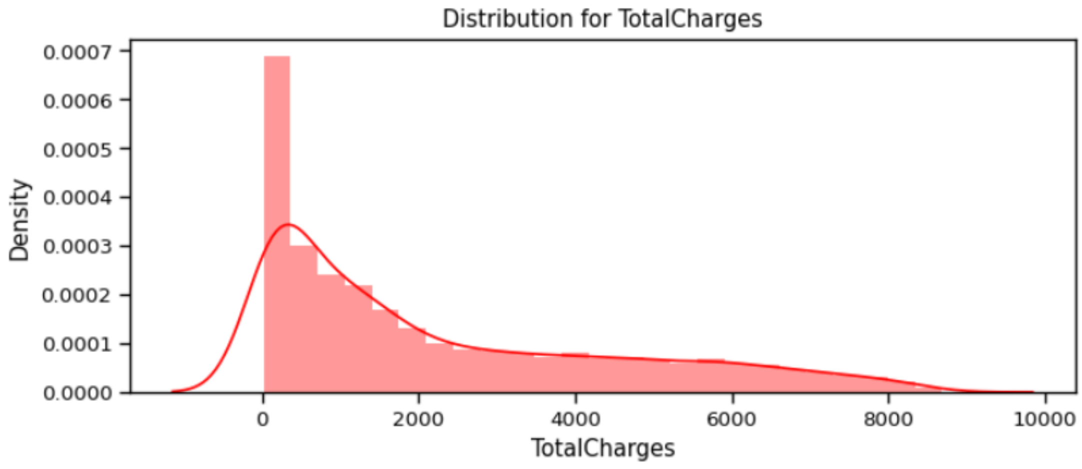


In [49]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(8,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color= color)
```

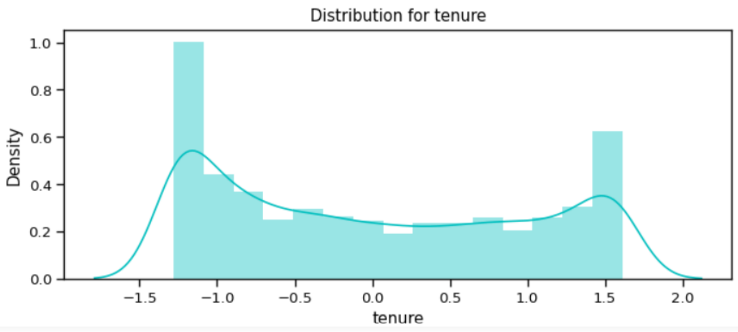
In [50]:
```
num_cols = ["tenure", 'MonthlyCharges', 'TotalCharges']
for feat in num_cols: distplot(feat, df)
```

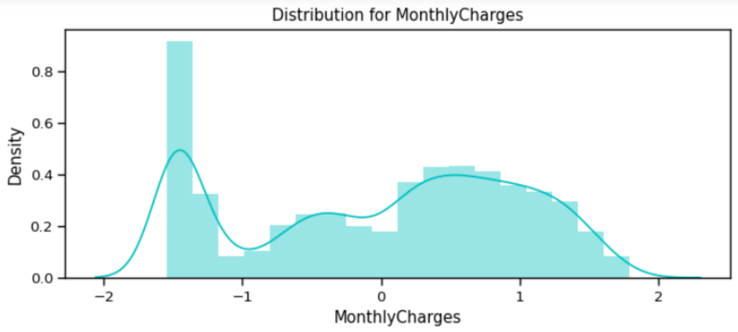


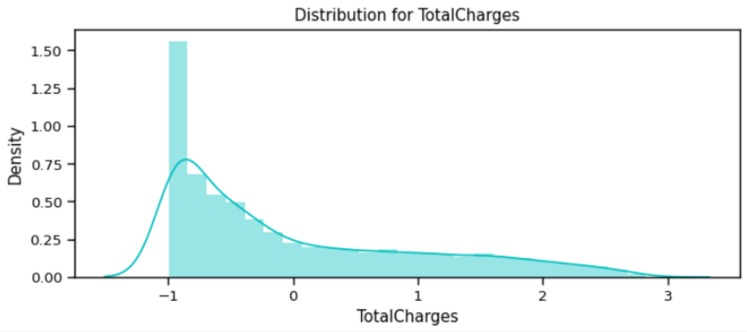




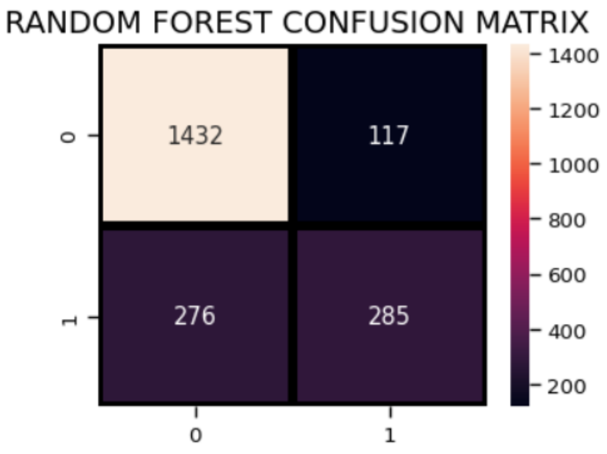
```
In [51]: df_std = pd.DataFrame(StandardScaler().fit_transform(df[num_cols].astype('float64')),
                               columns=num_cols)
         for feat in numerical_cols: distplot(feat, df_std, color='c')
```



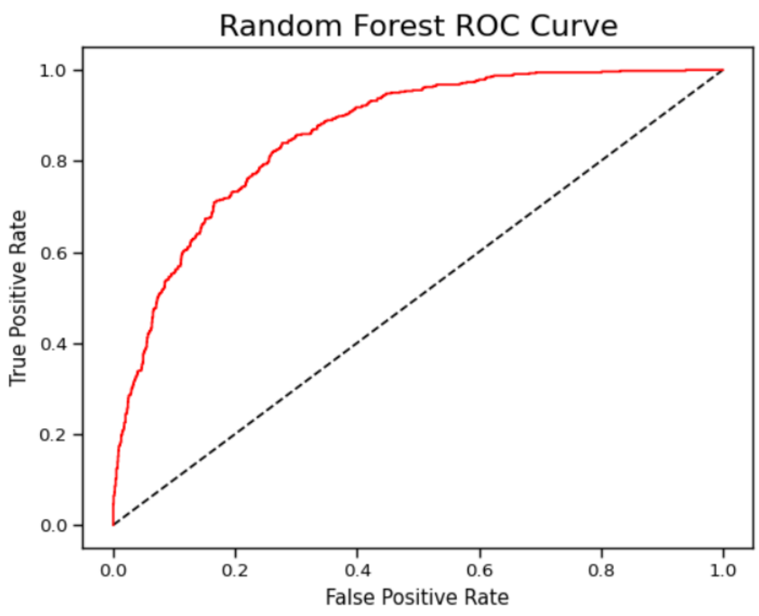




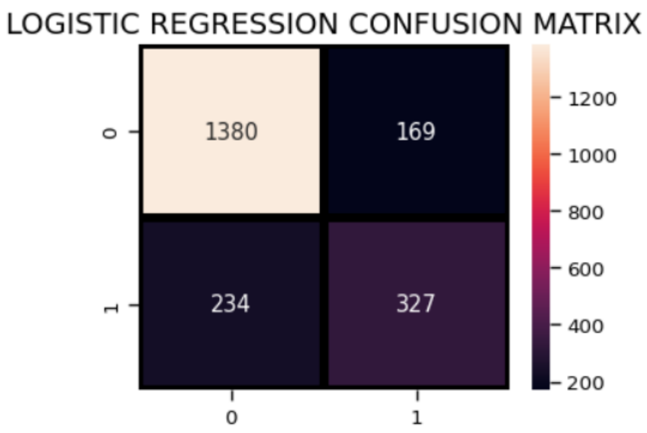
```
In [60]: plt.figure(figsize=(4,3))
         sns.heatmap(confusion_matrix(y_test, prediction_test),
                         annot=True,fmt = "d",linecolor="k",linewidths=3)

         plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
         plt.show()
```



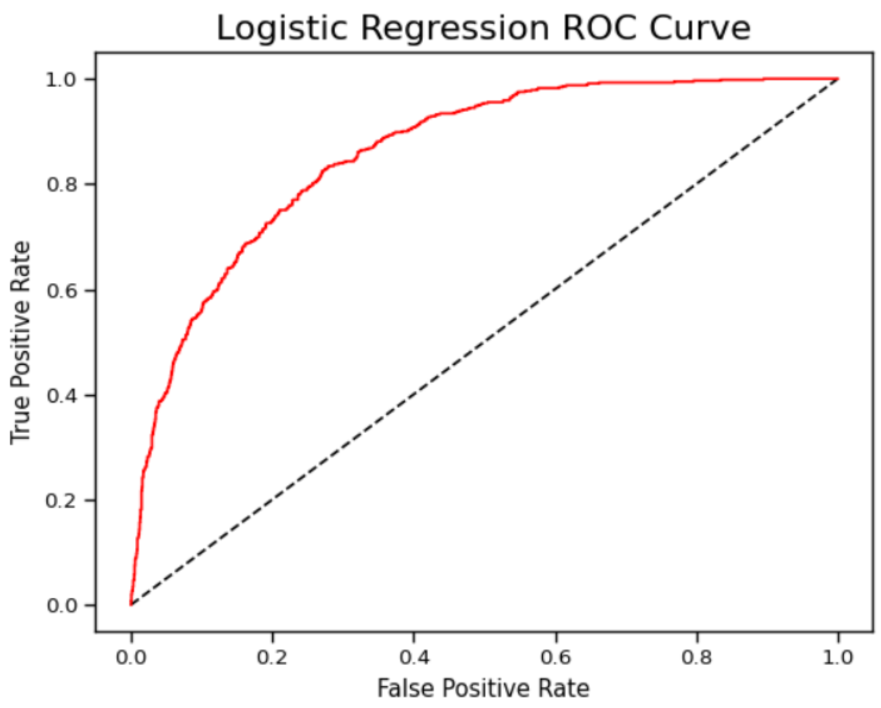
```
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



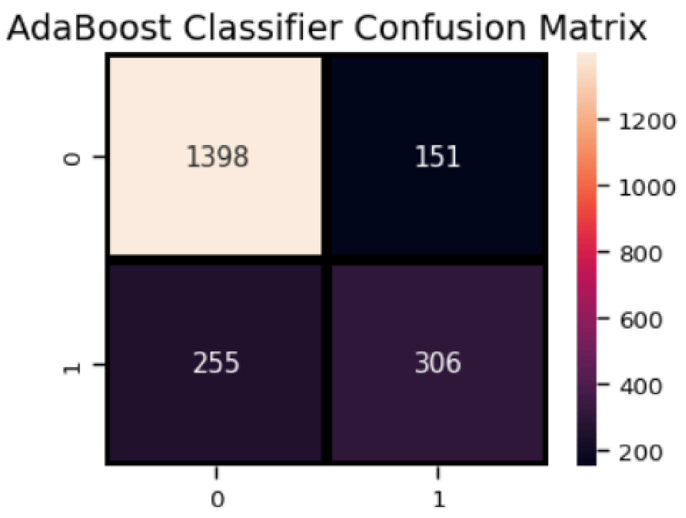
```
In [64]: plt.figure(figsize=(4,3))
         sns.heatmap(confusion_matrix(y_test, lr_pred),
                         annot=True,fmt = "d",linecolor="k",linewidths=3)

         plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)
         plt.show()
```



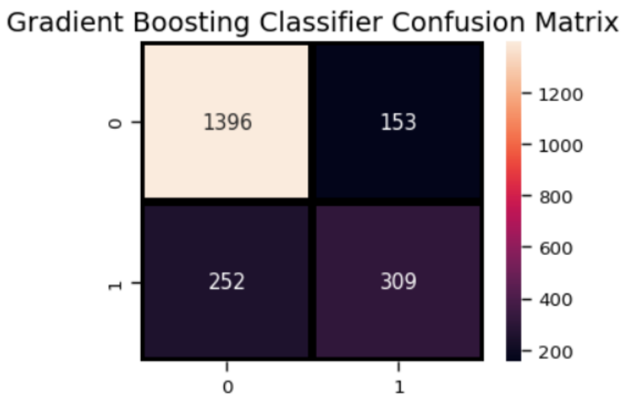
```
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```



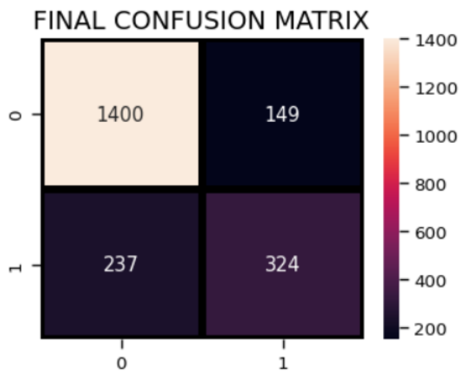
```
In [68]: plt.figure(figsize=(4,3))
         sns.heatmap(confusion_matrix(y_test, a_preds),
                         annot=True,fmt = "d",linecolor="k",linewidths=3)

         plt.title("AdaBoost Classifier Confusion Matrix",fontsize=14)
         plt.show()
```



In [71]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, gb_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("Gradient Boosting Classifier Confusion Matrix",fontsize=14)
plt.show()
```



In [74]:
```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test, predictions),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("FINAL CONFUSION MATRIX",fontsize=14)
plt.show()
```


FINAL CONFUSION MATRIX