In [53]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn import linear_model, preprocessing, metrics
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_au
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from platform import python_version
```

In [54]:

```
df = pd.read_csv('churn_clean.csv')
```

In [55]:

```
dfr = df[['Contract', 'Port_modem', 'Tablet', 'Phone', 'PaperlessBilling', 'InternetSe
```

In [56]:

```
dfr = pd.get_dummies(dfr, columns=['Contract', 'Port_modem', 'Tablet', 'Phone', 'Paper
```

In [57]:

```
dfr = dfr.rename(columns={'InternetService_Fiber Optic': 'FiberOptic', 'PaymentMethod_
```

In [58]:

```
dfr.drop('Churn_No', axis=1)
dfr.to_excel('Clean_Dataset_Task1.xlsx')
```

In [23]:

```
X = dfr[[col for col in dfr.columns if col != 'Churn_Yes']]
y = dfr['Churn_Yes']
```

In [24]:

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 28 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   MonthlyCharge            10000 non-null  float64
 1   Tenure                   10000 non-null  float64
 2   Children                 10000 non-null  int64  
 3   Age                      10000 non-null  int64  
 4   Outage_sec_perweek       10000 non-null  float64
 5   Email                    10000 non-null  int64  
 6   Contacts                 10000 non-null  int64  
 7   Yearly_equip_failure     10000 non-null  int64  
 8   Contract_Month-to-month  10000 non-null  uint8  
 9   ContractOneYear          10000 non-null  uint8  
 10  ContractTwoYear          10000 non-null  uint8  
 11  Port_modem_No            10000 non-null  uint8  
 12  Port_modem_Yes           10000 non-null  uint8  
 13  Tablet_No                10000 non-null  uint8  
 14  Tablet_Yes               10000 non-null  uint8  
 15  Phone_No                 10000 non-null  uint8  
 16  Phone_Yes                10000 non-null  uint8  
 17  PaperlessBilling_No      10000 non-null  uint8  
 18  PaperlessBilling_Yes     10000 non-null  uint8  
 19  InternetService_DSL      10000 non-null  uint8  
 20  FiberOptic               10000 non-null  uint8  
 21  InternetService_None     10000 non-null  uint8  
 22  Techie_No                10000 non-null  uint8  
 23  Techie_Yes               10000 non-null  uint8  
 24  AutoBankTransferPayment  10000 non-null  uint8  
 25  CreditCardPayment        10000 non-null  uint8  
 26  eCheckPayment            10000 non-null  uint8  
 27  MailedCheckPayment       10000 non-null  uint8  
dtypes: float64(3), int64(5), uint8(20)
memory usage: 820.4 KB
```

In [25]:

```
print(X.shape)
print(y.shape)
```

```
(10000, 28)
(10000,)
```

In [26]:

```
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
frames = [y, X]
df_std = pd.concat(frames, axis = 1)
print(df_std.head())
```

```
   Churn_Yes  MonthlyCharge    Tenure  Children       Age  Outage_sec_perweek  \
0          0      -0.003943 -1.048746 -0.972338  0.720925           -0.679978
1          1       1.630326 -1.262001 -0.506592 -1.259957            0.570331
2          0      -0.295225 -0.709940  0.890646 -0.148730            0.252347
3          0      -1.226521 -0.659524 -0.506592 -0.245359            1.650506
4          1      -0.528086 -1.242551 -0.972338  1.445638           -0.623156

      Email  Contacts  Yearly_equip_failure  Contract_Month-to-month  ...  \
0 -0.666282 -1.005852              0.946658                -1.095767  ...
1 -0.005288 -1.005852              0.946658                 0.912603  ...
2 -0.996779 -1.005852              0.946658                -1.095767  ...
3  0.986203  1.017588             -0.625864                -1.095767  ...
4  1.316700  1.017588              0.946658                 0.912603  ...

   PaperlessBilling_Yes  InternetService_DSL  FiberOptic  \
0              0.836721            -0.727842    1.126323
1              0.836721            -0.727842    1.126323
2              0.836721             1.373925   -0.887845
3              0.836721             1.373925   -0.887845
4             -1.195142            -0.727842    1.126323

   InternetService_None  Techie_No  Techie_Yes  AutoBankTransferPayment  \
0             -0.520083   0.449198   -0.449198                -0.535570
1             -0.520083  -2.226191    2.226191                 1.867168
2             -0.520083  -2.226191    2.226191                -0.535570
3             -0.520083  -2.226191    2.226191                -0.535570
4             -0.520083   0.449198   -0.449198                -0.535570

   CreditCardPayment  eCheckPayment  MailedCheckPayment
0           1.949556      -0.717421           -0.544993
1          -0.512937      -0.717421           -0.544993
2           1.949556      -0.717421           -0.544993
3          -0.512937      -0.717421            1.834888
4          -0.512937      -0.717421            1.834888

[5 rows x 29 columns]
```

In [27]:

```
#feature_names = X.columns
skbest = SelectKBest(score_func = f_classif, k='all')
X_new = skbest.fit_transform(X,y)
```

In [28]:

```
p_values = pd.DataFrame({'Feature':X.columns,
                        'p_value':skbest.pvalues_}).sort_values('p_value')
p_values[p_values['p_value']<.05]
```

Out[28]:

	Feature	p_value
0	MonthlyCharge	0.000000e+00
1	Tenure	0.000000e+00
8	Contract_Month-to-month	1.236727e-163
10	ContractTwoYear	3.019204e-72
9	ContractOneYear	2.359068e-44
19	InternetService_DSL	7.391267e-21
23	Techie_Yes	2.408802e-11
22	Techie_No	2.408802e-11
20	FiberOptic	4.873098e-09
21	InternetService_None	1.599912e-04
26	eCheckPayment	2.774461e-03
16	Phone_Yes	8.543973e-03
15	Phone_No	8.543973e-03

In [29]:

```
features_to_keep = p_values['Feature'][p_values['p_value']<.05]
features_to_keep
```

Out[29]:

```
0                MonthlyCharge
1                       Tenure
8      Contract_Month-to-month
10             ContractTwoYear
9              ContractOneYear
19         InternetService_DSL
23                  Techie_Yes
22                   Techie_No
20                  FiberOptic
21        InternetService_None
26               eCheckPayment
16                   Phone_Yes
15                    Phone_No
Name: Feature, dtype: object
```

In [30]:

```
X = X[['MonthlyCharge', 'Tenure', 'Contract_Month-to-month', 'ContractTwoYear', 'Contr
```

In [ ]:

```
dfr.drop('Churn_No', axis=1)
dfr.to_excel('Clean_Dataset_Task2.xlsx')
```

In [33]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state
```

In [34]:

```
X_train.to_excel('X_train.xlsx')
y_train.to_excel('y_train.xlsx')
X_test.to_excel('X_test.xlsx')
y_test.to_excel('y_test.xlsx')
```

In [35]:

```
knn = KNeighborsClassifier() #(n_neighbors=6)
knn.fit(X_train, y_train)
```

```
y_pred = knn.predict(X_test)
print(knn.predict(X_test))
```

```
[0 0 1 ... 0 0 1]
```

In [36]:

```
matrix = confusion_matrix(y_test, y_pred)
print(matrix)
```

```
[[1683  133]
 [ 161  523]]
```

In [37]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      1816
           1       0.80      0.76      0.78       684

    accuracy                           0.88      2500
   macro avg       0.85      0.85      0.85      2500
weighted avg       0.88      0.88      0.88      2500
```

In [38]:

```
print(knn.score(X_test, y_test))
```

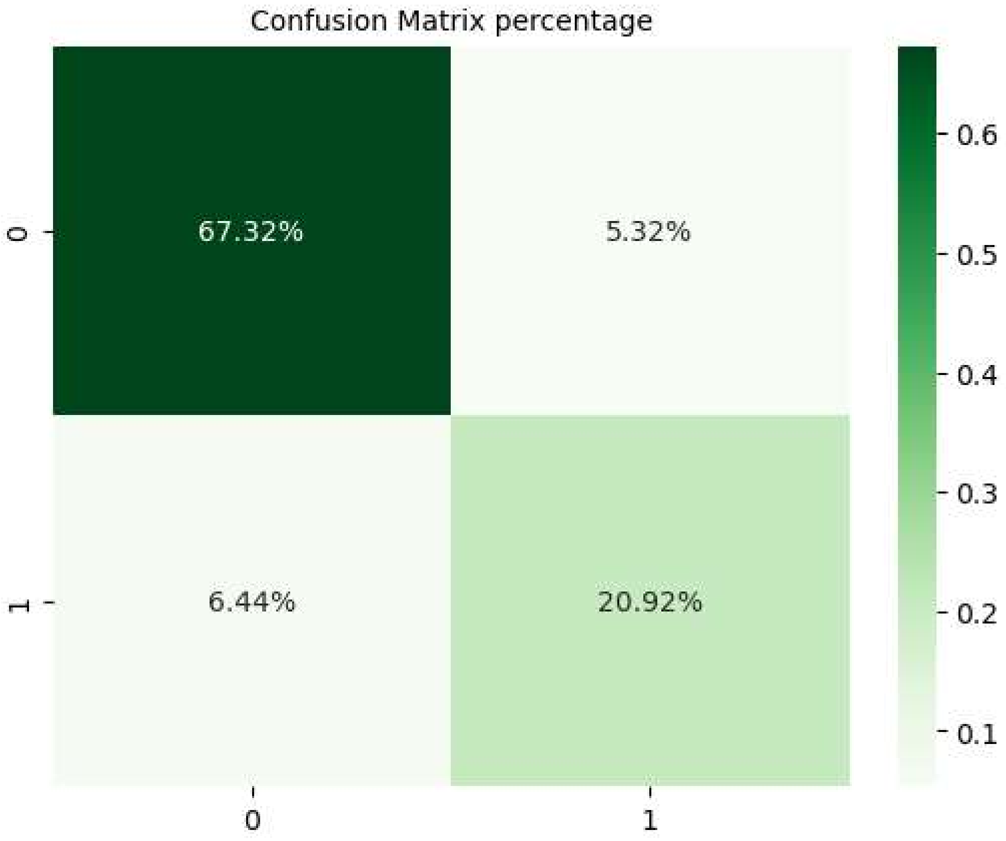
```
0.8824
```

In [39]:

```
sns.heatmap(matrix/np.sum(matrix), annot=True, fmt='.2%', cmap='Greens')
plt.title("Confusion Matrix percentage", fontsize =10)
```

Out[39]:

```
Text(0.5, 1.0, 'Confusion Matrix percentage')
```



In [40]:

```
total = matrix[0,0] + matrix[1,0] + matrix[0,1] + matrix[1,1]
accuracy = (matrix[0,0]+matrix[1,1])/total
```

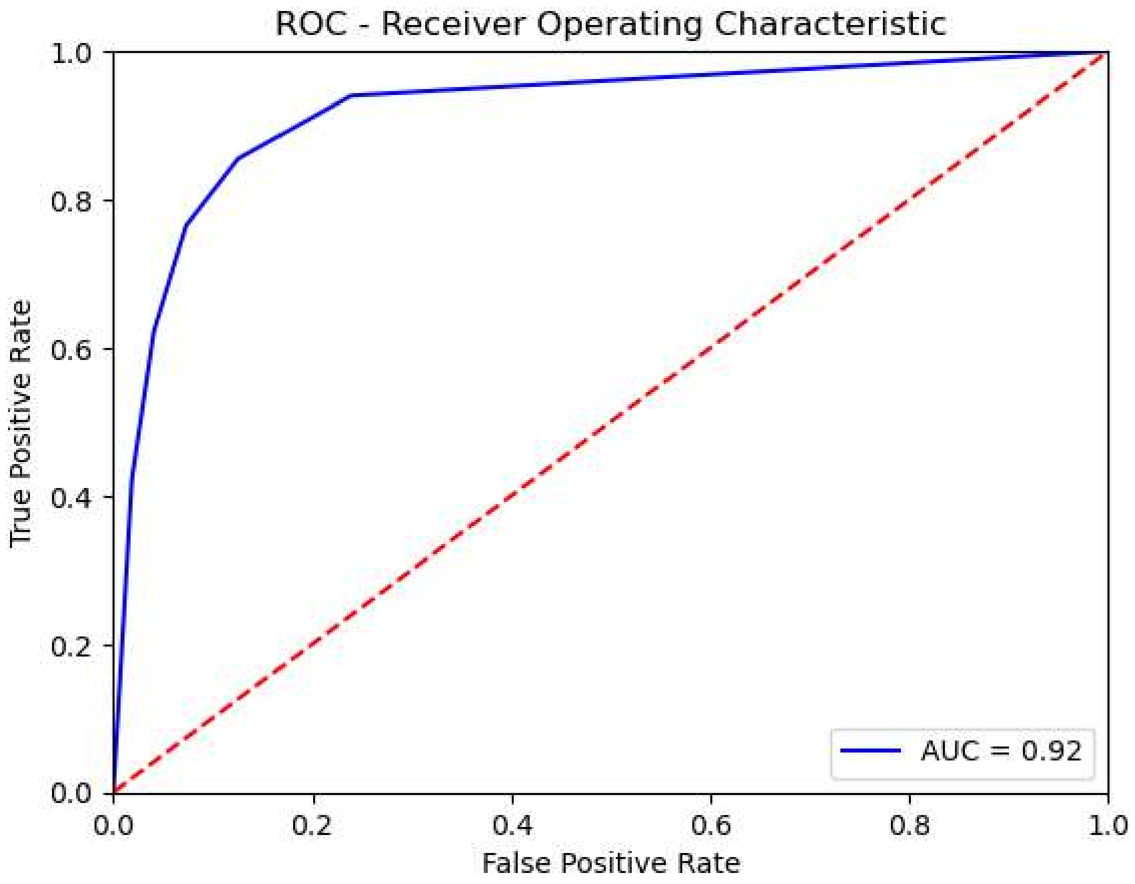
```
In [41]: print('Accuracy: {}'.format(accuracy))
```

```
Accuracy: 0.8824
```

```
In [42]: import sklearn.metrics as metrics
         probs = knn.predict_proba(X_test)
         preds = probs[:,1]
         fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
         roc_auc = metrics.auc(fpr, tpr)
         print(roc_auc)
```

```
0.920368330885952
```

```
In [43]: plt.title('ROC - Receiver Operating Characteristic')
         plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
         plt.legend(loc = 'lower right')
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0, 1])
         plt.ylim([0, 1])
         plt.ylabel('True Positive Rate')
         plt.xlabel('False Positive Rate')
         plt.show()
```



```
In [44]: python_version()
```

```
Out[44]: '3.9.13'
```

```
In [45]: !jupyter --version
```

```
Selected Jupyter core packages...
IPython          : 7.31.1
ipykernel        : 6.15.2
ipywidgets       : 7.6.5
jupyter_client   : 7.3.4
jupyter_core     : 4.11.1
jupyter_server   : 1.18.1
jupyterlab       : 3.4.4
nbclient         : 0.5.13
nbconvert        : 6.4.4
nbformat         : 5.5.0
notebook         : 6.4.12
qtconsole        : 5.2.2
traitlets        : 5.1.1
```

In [ ]:

```
pd.__version__
```

Out[ ]:

```
'1.4.4'
```

In [ ]:

```
np.__version__
```

Out[ ]:

```
'1.21.5'
```

In [ ]:

```
sklearn.__version__
```

Out[ ]:

```
'1.0.2'
```