In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from sklearn import metrics, tree
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, r
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from platform import python_version
```

In [2]:

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

In [3]:

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

In [4]:

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

In [5]:

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

In [6]:

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

In [7]:

```
X = dfr[[col for col in dfr.columns if col != 'Churn_Yes']]
y = dfr['Churn_Yes']
X = X.drop('Churn_No', axis=1)
```

In [8]:

```
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 [9]:

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

In [10]:

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

Out[10]:

	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
22	Techie_No	2.408802e-11
23	Techie_Yes	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 [11]:

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

Out[11]:

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

In [12]:

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

In [13]:

```
#X.to_excel('cleaned_DatasetT2.xlsx')
```

In [14]:

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

In [15]:

```
X_train.to_excel('X_trainT2.xlsx')
y_train.to_excel('y_trainT2.xlsx')
X_test.to_excel('X_testT2.xlsx')
y_test.to_excel('y_testT2.xlsx')
```

In [16]:

```
SEED = 1
dt = DecisionTreeClassifier(random_state=SEED)
dt.fit(X_train, y_train)
```

```
y_pred = dt.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[16]:

```
0.8552
```

In [17]:

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

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90      1816
           1       0.74      0.73      0.73       684

    accuracy                           0.86      2500
   macro avg       0.82      0.82      0.82      2500
weighted avg       0.85      0.86      0.85      2500
```

In [18]:

```
mse_dt = MSE(y_test, y_pred)
print("MSE:", mse_dt)
```

```
MSE: 0.1448
```

In [19]:

```
params_dt = {'max_depth': [2,3,4,5,6]} #, 'max_features':[.02, .04, .06, .07]
```

In [20]:

```
grid_dt = GridSearchCV(estimator=dt, param_grid=params_dt, cv=5, n_jobs=-1)
```

In [21]:

```
grid_dt.fit(X_train, y_train)
```

Out[21]:

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=1), n_jobs=-1,
             param_grid={'max_depth': [2, 3, 4, 5, 6]})
```

In [22]:

```
print(grid_dt.best_score_, grid_dt.best_params_)
```

```
0.8779999999999999 {'max_depth': 6}
```

In [23]:

```
dt = DecisionTreeClassifier(max_depth=6, random_state=SEED)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[23]:

```
0.8764
```

In [24]:

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

```
              precision    recall  f1-score   support

           0       0.92      0.91      0.91      1816
           1       0.76      0.79      0.78       684

    accuracy                           0.88      2500
   macro avg       0.84      0.85      0.85      2500
weighted avg       0.88      0.88      0.88      2500
```

In [25]:

```
print("MSE:", mse_dt)
```

```
MSE: 0.1448
```

In [26]:

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

```
print(matrix)
```

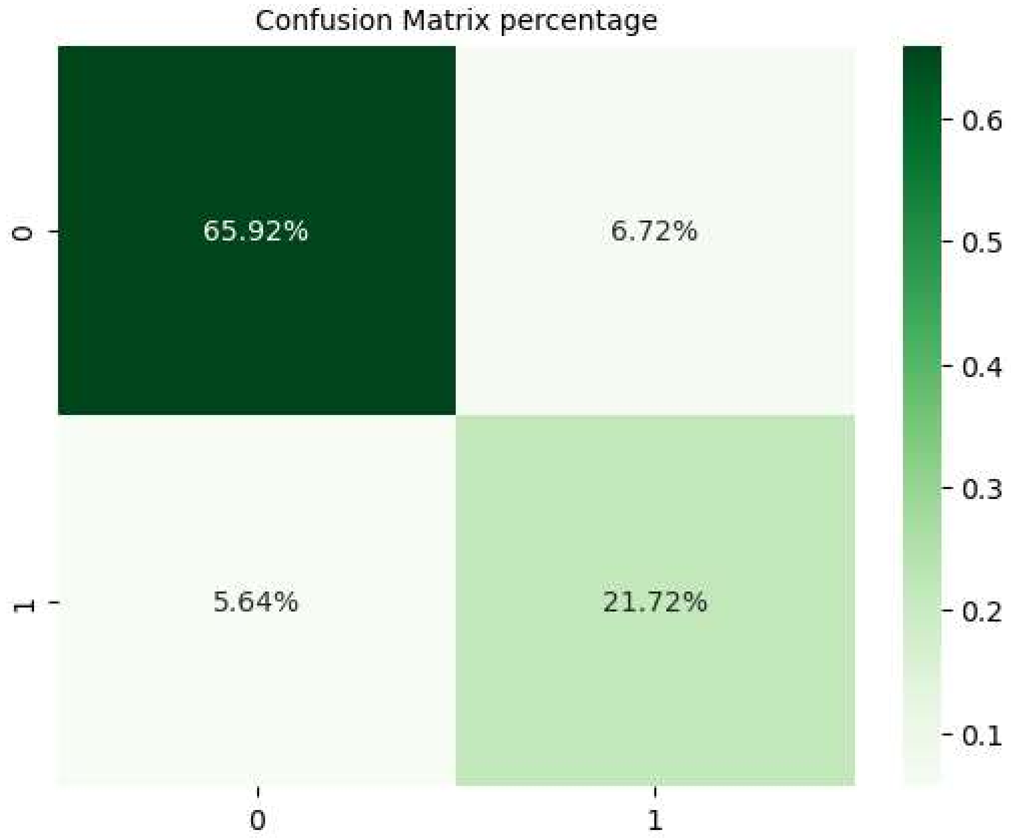
```
[[1648  168]
 [ 141  543]]
```

In [27]:

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

Out[27]:

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



In [28]:

```
python_version()
```

Out[28]:

```
'3.9.13'
```

In [29]:

```
!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 [30]:

```
pd.__version__
```

```
Out[30]: '1.4.4'
```

```
In [31]: np.__version__
```

```
Out[31]: '1.21.5'
```

```
In [32]: sklearn.__version__
```

```
Out[32]: '1.0.2'
```

```
In [ ]:
```