In [1]:

```
#Import library yang dibutuhkan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import pickle
from pathlib import Path
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: p
andas.util.testing is deprecated. Use the functions in the public API at pandas.testing i
nstead.
  import pandas.util.testing as tm
```

In [2]:

```
#import dataset
df_load = pd.read_csv('https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/dqlab_telco_
final.csv')

#Tampilkan bentuk dari dataset
print(df_load.shape)

#Tampilkan 5 data teratas
print(df_load.head())

#Tampilkan jumlah ID yang unik
print(df_load.customerID.nunique())
```

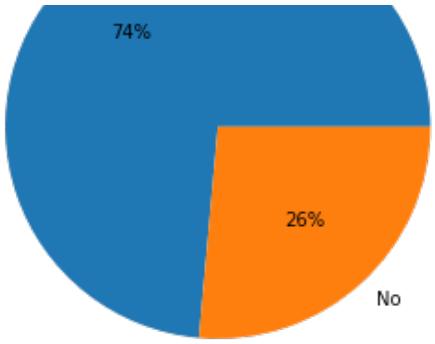
```
(6950, 13)
   UpdatedAt   customerID  gender  ... MonthlyCharges TotalCharges  Churn
0     202006  45759018157  Female  ...          29.85        29.85     No
1     202006  45315483266    Male  ...          20.50      1198.80     No
2     202006  45236961615    Male  ...         104.10       541.90    Yes
3     202006  45929827382  Female  ...         115.50      8312.75     No
4     202006  45305082233  Female  ...          81.25      4620.40     No

[5 rows x 13 columns]
6950
```

In [3]:

```
#import matplotlib dan seaborn

import matplotlib.pyplot as plt
import seaborn as sns
```

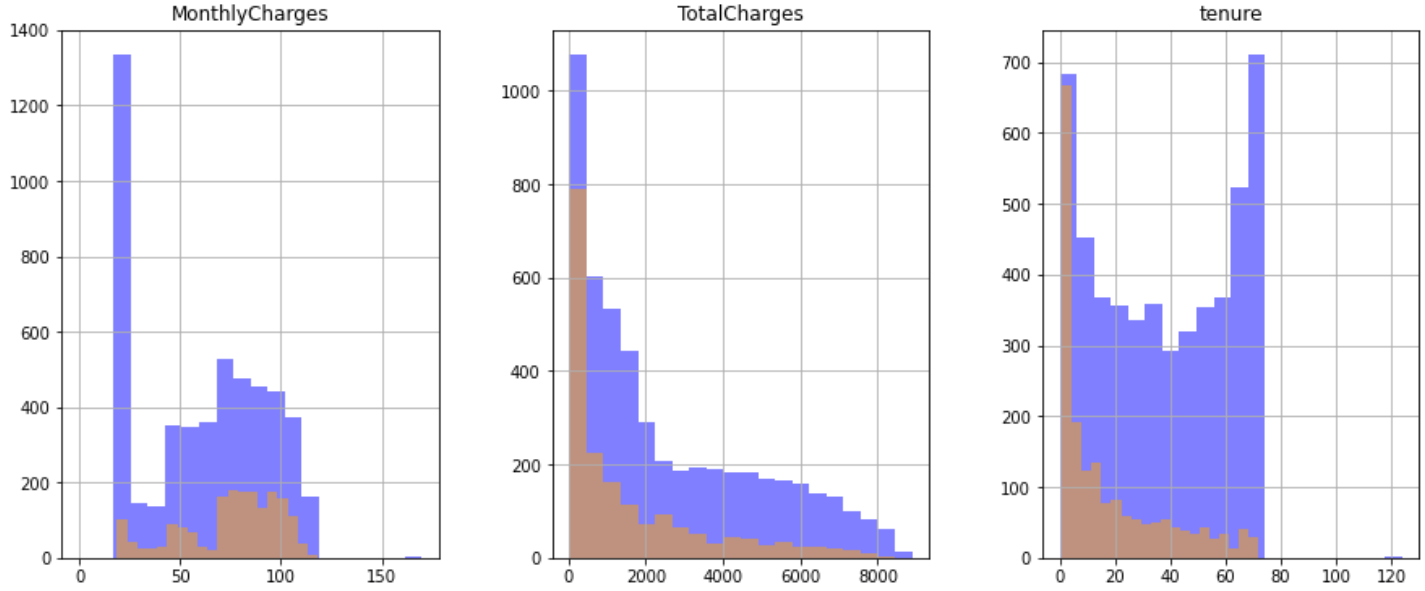
In [4]:

```
from matplotlib import pyplot as plt
import numpy as np
#Your codes here
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
labels = ['Yes','No']
churn = df_load.Churn.value_counts()
ax.pie(churn, labels=labels, autopct='%.0f%%')
plt.show()
```



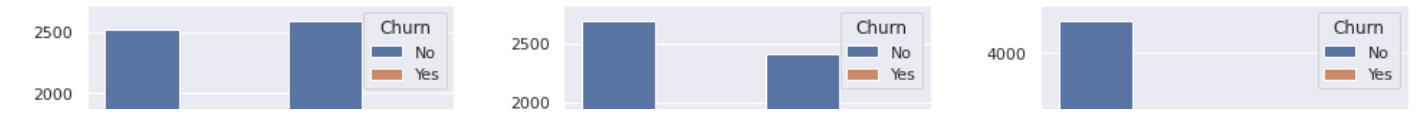
In [5]:

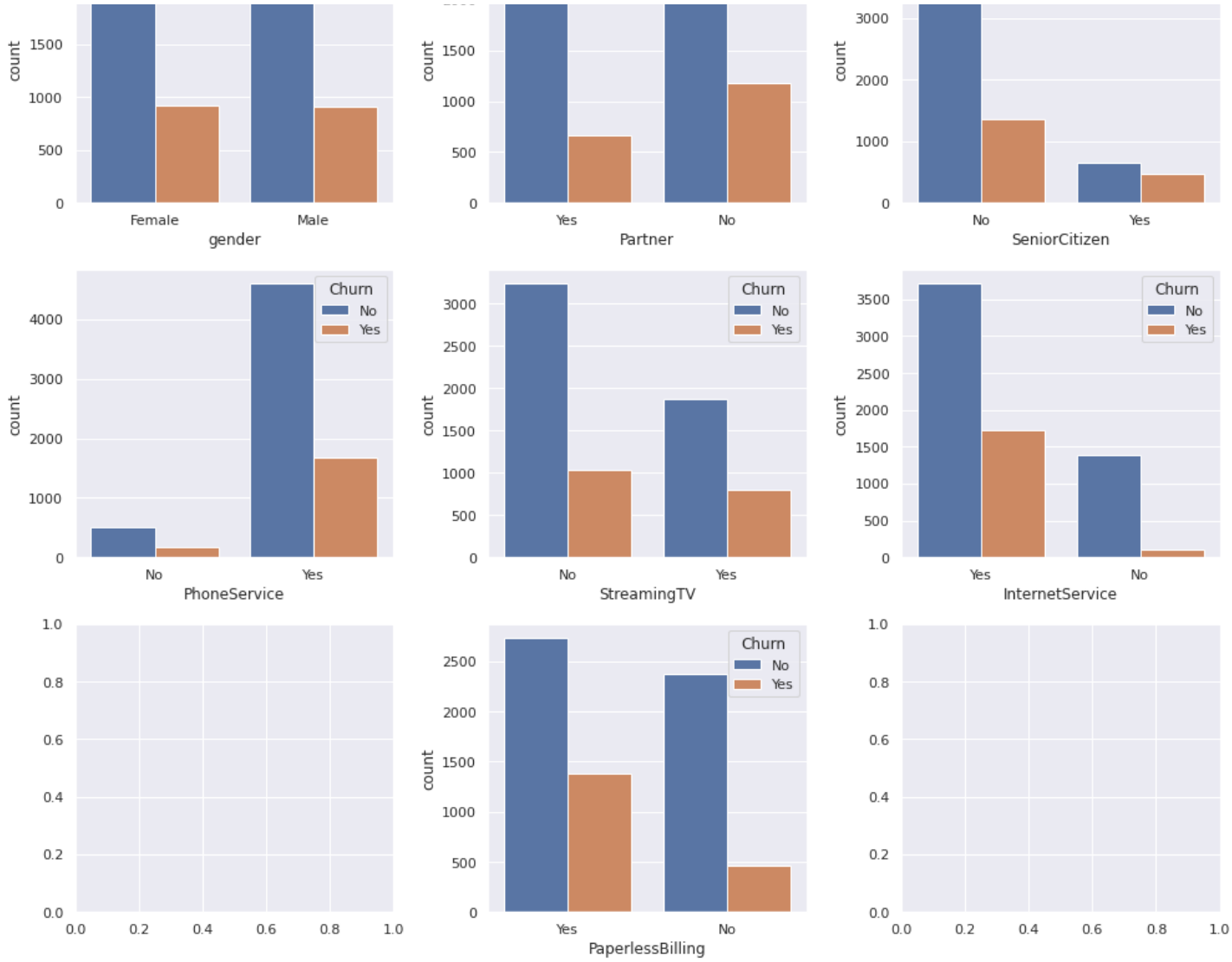
```
from matplotlib import pyplot as plt
import numpy as np

#creating bin in chart
numerical_features = ['MonthlyCharges','TotalCharges','tenure']
fig, ax = plt.subplots(1, 3, figsize=(15, 6))
# Use the following code to plot two overlays of histogram per each numerical_features, u
se a color of blue and orange, respectively
df_load[df_load.Churn == 'No'][numerical_features].hist(bins=20, color='blue', alpha=0.5
, ax=ax)
df_load[df_load.Churn == 'Yes'][numerical_features].hist(bins=20, color='orange', alpha=
0.5, ax=ax)
plt.show()
```



In [6]:

```
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
sns.set(style='darkgrid')
# Your code goes here
fig, ax = plt.subplots(3, 3, figsize=(14, 12))
sns.countplot(data=df_load, x='gender', hue='Churn', ax=ax[0][0])
sns.countplot(data=df_load, x='Partner', hue='Churn', ax=ax[0][1])
sns.countplot(data=df_load, x='SeniorCitizen', hue='Churn', ax=ax[0][2])
sns.countplot(data=df_load, x='PhoneService', hue='Churn', ax=ax[1][0])
sns.countplot(data=df_load, x='StreamingTV', hue='Churn', ax=ax[1][1])
sns.countplot(data=df_load, x='InternetService', hue='Churn', ax=ax[1][2])
sns.countplot(data=df_load, x='PaperlessBilling', hue='Churn', ax=ax[2][1])
plt.tight_layout()
plt.show()
```





In [7]:

```
#Remove the unnecessary columns customerID & UpdatedAt
cleaned_df = df_load.drop(['customerID','UpdatedAt'], axis=1)
print(cleaned_df.head())
```

```
   gender SeniorCitizen Partner  ...  MonthlyCharges TotalCharges Churn
0  Female             No     Yes  ...           29.85        29.85    No
1    Male             No     Yes  ...           20.50      1198.80    No
2    Male             No      No  ...          104.10       541.90   Yes
3  Female             No     Yes  ...          115.50      8312.75    No
4  Female             No     Yes  ...           81.25      4620.40    No

[5 rows x 11 columns]
```

In [8]:

```
from sklearn.preprocessing import LabelEncoder
#Convert all the non-numeric columns to numerical data types
for column in cleaned_df.columns:
 if cleaned_df[column].dtype == np.number: continue
 # Perform encoding for each non-numeric column
 cleaned_df[column] = LabelEncoder().fit_transform(cleaned_df[column])
print(cleaned_df.describe())
```

```
            gender  SeniorCitizen  ...  TotalCharges        Churn
count  6950.000000    6950.000000  ...   6950.000000  6950.000000
mean      0.504317       0.162302  ...   2286.058750     0.264173
std       0.500017       0.368754  ...   2265.702553     0.440923
min       0.000000       0.000000  ...     19.000000     0.000000
25%       0.000000       0.000000  ...    406.975000     0.000000
50%       1.000000       0.000000  ...   1400.850000     0.000000
75%       1.000000       0.000000  ...   3799.837500     1.000000
max       1.000000       1.000000  ...   8889.131250     1.000000
```

```
[8 rows x 11 columns]
```

In [9]:

```
from sklearn.model_selection import train_test_split
# Predictor dan target
X = cleaned_df.drop('Churn', axis = 1)
y = cleaned_df['Churn']
# Splitting train and test
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
)
# Print according to the expected result
print('Jumlah baris dan kolom dari x_train adalah:', x_train.shape,', sedangkan Jumlah ba
ris dan kolom dari y_train adalah:', y_train.shape)
print('Prosentase Churn di data Training adalah:')
print(y_train.value_counts(normalize=True))
print('Jumlah baris dan kolom dari x_test adalah:', x_test.shape,', sedangkan Jumlah bar
is dan kolom dari y_test adalah:', y_test.shape)
print('Prosentase Churn di data Testing adalah:')
print(y_test.value_counts(normalize=True))
```

```
Jumlah baris dan kolom dari x_train adalah: (4865, 10) , sedangkan Jumlah baris dan kolom
dari y_train adalah: (4865,)
Prosentase Churn di data Training adalah:
0    0.734841
1    0.265159
Name: Churn, dtype: float64
Jumlah baris dan kolom dari x_test adalah: (2085, 10) , sedangkan Jumlah baris dan kolom
dari y_test adalah: (2085,)
Prosentase Churn di data Testing adalah:
0    0.738129
1    0.261871
Name: Churn, dtype: float64
```

Logistic

Membuat model Klasifikasi menggunakan LogisticRegression

In [10]:

```
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression().fit(x_train, y_train)
print('Model Logistic Regression yang terbentuk adalah: \n', log_model)
```

```
Model Logistic Regression yang terbentuk adalah:
 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Model Training Logistic Regression

In [11]:

```
from sklearn.metrics import classification_report
# Predict
y_train_pred = log_model.predict(x_train)
# Print classification report
```

```
print('Classification Report Training Model (Logistic Regression) :')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Logistic Regression) :
              precision    recall  f1-score   support

           0       0.83      0.90      0.87      3575
           1       0.65      0.49      0.56      1290

    accuracy                           0.80      4865
   macro avg       0.74      0.70      0.71      4865
weighted avg       0.78      0.80      0.79      4865
```

Model Testing Logistic Regression

In [12]:

```
from sklearn.metrics import classification_report
# Predict
y_test_pred = log_model.predict(x_test)
# Print classification report
print('Classification Report Testing Model (Logistic Regression):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Logistic Regression):
              precision    recall  f1-score   support

           0       0.83      0.90      0.87      1539
           1       0.64      0.48      0.55       546

    accuracy                           0.79      2085
   macro avg       0.73      0.69      0.71      2085
weighted avg       0.78      0.79      0.78      2085
```

Decision Tree Classifier

Membuat model Klasifikasi menggunakan Decision Tree Classifier

In [13]:

```
from sklearn.tree import DecisionTreeClassifier
tree_model = DecisionTreeClassifier().fit(x_train, y_train)
print('Model Decision Tree Classifier yang terbentuk adalah: \n', tree_model)
```

```
Model Decision Tree Classifier yang terbentuk adalah:
 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Model Training Decision Tree

In [14]:

```
from sklearn.metrics import classification_report
# Predict
y_train_pred = tree_model.predict(x_train)
# Print classification report
print('Classification Report Training Model (Decision Tree) :')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Decision Tree) :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      3575
```

```
           1       1.00      0.98      0.99      1290

    accuracy                           1.00      4865
   macro avg       1.00      0.99      0.99      4865
weighted avg       1.00      1.00      1.00      4865
```

Model Testing Decision Tree

In [15]:

```
from sklearn.metrics import classification_report
# Predict
y_test_pred = tree_model.predict(x_test)
# Print classification report
print('Classification Report Testing Model (Decision Tree):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Decision Tree):
              precision    recall  f1-score   support

           0       0.82      0.81      0.81      1539
           1       0.48      0.49      0.48       546

    accuracy                           0.73      2085
   macro avg       0.65      0.65      0.65      2085
weighted avg       0.73      0.73      0.73      2085
```

Naive Bayes Classifier

Membuat model Klasifikasi menggunakan Naive Bayes Classifier

In [16]:

```
from sklearn.naive_bayes import GaussianNB
gnb_model = GaussianNB().fit(x_train, y_train)
print('Model Naive Bayes Classifier yang terbentuk adalah: \n', gnb_model)
```

```
Model Naive Bayes Classifier yang terbentuk adalah:
 GaussianNB(priors=None, var_smoothing=1e-09)
```

Model Training Naive Bayes

In [17]:

```
from sklearn.metrics import classification_report
# Predict
y_train_pred = gnb_model.predict(x_train)
# Print classification report
print('Classification Report Training Model (Naive Bayes) :')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Naive Bayes) :
              precision    recall  f1-score   support

           0       0.88      0.75      0.81      3575
           1       0.51      0.72      0.60      1290

    accuracy                           0.74      4865
   macro avg       0.70      0.73      0.70      4865
weighted avg       0.78      0.74      0.76      4865
```

Model Testing Naive Bayes

In [18]:

```
from sklearn.metrics import classification_report
# Predict
y_test_pred = gnb_model.predict(x_test)
# Print classification report
print('Classification Report Testing Model (Naive Bayes):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Naive Bayes):
              precision    recall  f1-score   support

           0       0.87      0.75      0.80      1539
           1       0.49      0.67      0.56       546

    accuracy                           0.73      2085
   macro avg       0.68      0.71      0.68      2085
weighted avg       0.77      0.73      0.74      2085
```

Ringkasan Model

1. **Logistic Regression (akurasi training 80%, akurasi testing 79%)**
2. **Decision Tree Classifier (akurasi training 100%, akurasi testing 73%)**
3. **Naive Bayes Classifier (akurasi training 74%, akurasi testing 73%)**

Kesimpulan

Berdasarkan hasil dan ringkasa model, dapat disimpulkan bahwa model dengan akurasi terbaik dimiliki oleh Logistic Regression. Hal ini dikarenakan performa dari model Logistic Regression cenderung mampu memprediksi sama baiknya di fase training maupun testing.

Akan tetapi hal ini tidak menjadikan kita untuk menarik kesimpulan bahwsannya jika untuk melakukan pemodelan apapun maka digunakan Logistic Regression, kita tetap harus melakukan banyak percobaan model untuk menentukan mana yang terbaik.

In [ ]:

```
```

In [ ]:

```
```