Tutorial 2 for Chapter 2

Case study 3: Contract Renewal Prediction by KNN and Naive Bayesian models

Reference: Python数据挖掘实战

For the course AMA546 Statistical Data Mining
Lecturer: Dr. Catherine Liu
AMA, PolyU, HKSAR

Contents

In [1]:

```
import pandas
# sklearn = Scikit-learn official website: https://scikit-learn.org/stable/index.html
from sklearn.model_selection import train_test_split # Import the training and testing dataset splitter
from sklearn.neighbors import KNeighborsClassifier # Import the KNN classifier
from sklearn.naive_bayes import GaussianNB # Import Gussian native bayesian classifier
from sklearn.metrics import confusion_matrix # Import the confusion matrix generater
from sklearn.metrics import accuracy_score # Import the accuracy generater
from sklearn.metrics import precision_score # Import the precision generater
from sklearn.metrics import recall_score # Import the recall generater
from sklearn.metrics import f1_score # Import the f1 generater
from sklearn.model_selection import cross_val_score # Import the cross validation score generater
from sklearn.metrics import make_scorer # Import make_scorer module to make a scorer from a loss function
import matplotlib.pyplot as plt # Import plot module
```

executed in 1.83s, finished 23:53:31 2023-02-13

1 Objectives of the analysis

In this case study, we employ both the KNN (K Nearest Neighbors) model and the Native Bayes Classifier to predict the likelihood of a merchant renewing their contracts.

The purpose of this study is to aid a store leasing company in South China in developing a predictive model based on their merchants data to anticipate the probability of merchants in North China renewing their contracts. This information is invaluable to the company's business department, serving as a foundation for their future investment planning.

The historical research has revealed that the key factors influencing a merchant's decision to renew their contract are the **duration of registration**, **revenue**, and **rental cost**. Furthermore, the circumstances in South China are comparable to those in North China, making the model developed using data from South China applicable to the situation in North China.

2 Description of the data

We have two CSV (Comma Separated Values) files at our disposal, which contain the data collected from South China and North China respectively. Before we proceed any further, let us first load the data.

In [2]:

```
### Load data
# data from the South China
data_south = pandas.read_csv(
    'South_China.csv',
    encoding='utf8'  # the csv file is encoding as UTF-8
)
# the data from North Chinad
data_north = pandas.read_csv(
    'North_China.csv',
    encoding='utf8'
)
```

executed in 7ms, finished 23:53:31 2023-02-13

2.1 Preview data from the South China

In [3]:

```
### View data from the South China
print('The shape of the data is:', data_south.shape)
data_south.head()
```

executed in 7ms, finished 23:53:31 2023-02-13

```
The shape of the data is: (1500, 5)
```

Out[3]:

	ID	Registration_Duration	Revenue	Cost	Renewal
0	10001	29	46	8.33316	True
1	10002	58	105	18.24564	True
2	10003	36	55	3.13296	True
3	10004	32	28	9.96705	True
4	10005	26	32	7.19040	False

2.2 Preview data from the North China

In [4]:

```
### View data from the North China
print('The shape of the data is:', data_north.shape)
data_north.head()
```

executed in 4ms, finished 23:53:31 2023-02-13

```
The shape of the data is: (325, 4)
```

Out[4]:

	ID	Registration_Duration	Revenue	Cost
0	20001	29	46	8.33316
1	20002	26	32	7.19040
2	20003	59	172	10.11150
3	20004	22	24	0.95979
4	20005	56	87	17.97600

2.3 Brief Overview

The `data_south` dataset serves as our training data and includes a total of **1500 rows (samples)**, each with **5 columns**: `ID`, `Registration_Duration`, `Revenue`, `Cost`, and `Renewal`. Each merchant is assigned a unique `ID` to track their information. The `Registration_Duration` is the length of time the merchant has been registered with the company, measured in months. The `Revenue` column records the monthly earnings generated by a merchant. The `Cost` represents the rental cost paid by the merchant, recorded in thousands of dollars. The `Renewal` variable is a binary outcome indicating whether the merchant decided to renew its contract or not, represented with either `TRUE` or `FALSE`.

The `data_north` dataset contains **325 samples** with **4 columns**.**The missing column,** **`Renewal`**, is what needs to be predicted. The variables in this dataset are used for prediction purposes.

The table below details the variables we can use:

Variable	Explanation	Range (data_south)	Range (data_nouth)
ID	Unique identifier for each merchant	10001, 10002, ..., 11500	20001, 20002, ..., 20325
Registration_Duration	Duration of registration in months	[19,72]	[19, 68]
Revenue	Revenue of the merchant in thousands of dollars	[13, 981]	[13, 448]
Cost	Rental cost of the merchant in thousands of dollars	[0.5136, 96.4712]	[0.5136, 71.80128]
Renewal	Whether the merchant decided to renew the contract or not (TRUE or FALSE)	TRUE or FALSE	Need to be predict

3 Exploratory data analysis

3.1 Data quality (No need in our case)

Before embarking on the data analysis, it is crucial to assess the quality of the dataset. In data quality assessment, we typically perform the following tasks:

1. Data Cleaning (https://en.wikipedia.org/wiki/Data_cleansing): This involves removing any missing or irrelevant data from the dataset.
2. Data Validation (https://en.wikipedia.org/wiki/Data_validation): This step involves checking the data for consistency and validity by comparing it with other observations (outlier detection) or data from other sources.
3. Data Transformation (https://en.wikipedia.org/wiki/Data_transformation_(computing)): This involves converting the data into a format that is suitable for analysis. This may include changing data types, data rescaling, or generating new variables.

Overall, the goal is to ensure that the data is **accurate**, **reliable**, and **suitable** for the subsequent use.

3.1.1 Data Cleaning

We will perform data cleaning by checking for any **`NA`** or **`Null`** values in our dataset. This can be easily done in Python by using the `isna()` * method. Fortunately, our dataset is free of any such values and does not require any cleaning in this regard.

* *`.isnull()` is an alias for `.isna()`. Both methods serve the same purpose, so we only need to use one of them to check for missing values.*

In [5]:

```
### There is no NA or Null data in the dataframe
for i in [data_north,data_south]:
    print(i[i.isna().any(axis=1)])
    i.isna()
```

executed in 3ms, finished 23:53:31 2023-02-13

```
Empty DataFrame
Columns: [ID, Registration_Duration, Revenue, Cost]
Index: []
Empty DataFrame
Columns: [ID, Registration_Duration, Revenue, Cost, Renewal]
Index: []
```

3.1.2 Data Validation

Additionally, we need to assess if there are any outliers in the observations. Outliers are data points that differ significantly from the rest of the data and can either be due to errors in data collection or reveal unique characteristics of the data. It is crucial to identify and analyze these outliers to ensure that our data analysis is both precise and comprehensive.

In dataset `data_south` :

- the **`Registration_Duration`** ranges from 19 to 72 months, while
- the **`Revenue`** ranges from 13,000 to 981,000 and the cost ranges from 513 to 96,471.
- In column **`Renewal`** , the sample exhibits a balanced distribution. The proportion of merchants who declined to renew their contract is 36.5%, while 63.5% of the merchants decide to renew their contract.

All of the data appears to be within reasonable bounds and no inconsistencies are immediately noticeable.

In [6]:

```
### Descriptive statistics
data_south.describe()
```

executed in 8ms, finished 23:53:31 2023-02-13

Out[6]:

	ID	Registration_Duration	Revenue	Cost
count	1500.000000	1500.000000	1500.000000	1500.000000
mean	10750.500000	33.756000	58.387333	6.191050
std	433.157015	10.928133	61.862056	8.206847
min	10001.000000	19.000000	13.000000	0.513600
25%	10375.750000	25.000000	28.000000	1.821140
50%	10750.500000	33.000000	40.000000	3.735905
75%	11125.250000	40.000000	64.000000	7.245237
max	11500.000000	72.000000	981.000000	96.471200

In [7]:

```
# Use crosstab to get the proportion of renewals
renewal_counts = pandas.crosstab(index=data_south["Renewal"],
                                 columns="Proportion",
                                 normalize='all') *100
round(renewal_counts,1).astype(str).apply(lambda x:x + '%')
```

executed in 8ms, finished 23:53:31 2023-02-13

Out[7]:

col_0	Proportion
Renewal	
False	36.5%
True	63.5%

In dataset `data_nouth` :

- the **`Registration_Duration`** ranges from 19 to 68 months, while
- the **`Revenue`** ranges from 13,000 to 448,000 and
- the **`Cost`** ranges from 513 to 71,801.

The same as dataset `data_south` , no anomalies are evident.

In [8]:

```
# Descriptive statistics
data_north.describe()
```

executed in 8ms, finished 23:53:31 2023-02-13

Out[8]:

	ID	Registration_Duration	Revenue	Cost
count	325.000000	325.000000	325.000000	325.000000
mean	20163.000000	33.833846	56.769231	6.200387
std	93.963645	10.973738	51.103311	7.912646
min	20001.000000	19.000000	13.000000	0.513600
25%	20082.000000	25.000000	28.000000	1.737680
50%	20163.000000	33.000000	40.000000	3.707550
75%	20244.000000	40.000000	64.000000	7.549920
max	20325.000000	68.000000	448.000000	71.801280

3.1.3 Data Transformation

All the data are stored in double format. The `Registration_Duration` is measured in months and the `Revenue` and `Cost` are measured in thousands of dollars. There is no need for any data type conversion or data rescaling.

3.2 Marginal variable analysis

The marginal variable analysis is only with respect to the dataset `data_south` .

3.2.1 Registration_Duration

The boxplot demonstrates that customers who decide to renew their contracts have higher lower quartile, median, upper quartile, and upper whisker values compared to those who do not renew their contracts. This indicates that customers who have been registered for a longer period of time are more likely to renew their contracts, while those who have been registered for a shorter time are more likely to discontinue their contracts. This substantial difference highlights the potential significance of the 'Registration_Duration' variable in explaining contract renewal behavior.

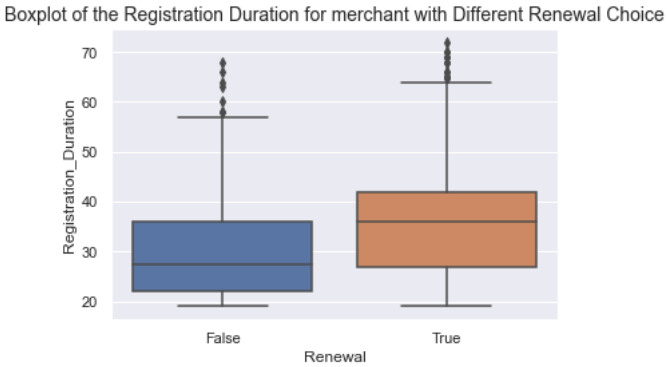
In [9]:

```
### Boxplot
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="ticks")
sns.set_style("darkgrid")
bp = sns.boxplot(x="Renewal", y="Registration_Duration", data=data_south)
bp.set_title("Boxplot of the Registration Duration for merchant with Different Renewal Choice",
             fontsize=14)
```

executed in 129ms, finished 23:53:31 2023-02-13

Out[9]:

```
Text(0.5, 1.0, 'Boxplot of the Registration Duration for merchant with Different Renewal Choice')
```



In [10]:

```
# ### violin plot
# data_south["all"] = ""
# ax = sns.violinplot(x="all", y="Registration_Duration", hue="Renewal", data=data_south, split=True)
```

executed in 1ms, finished 23:53:31 2023-02-13

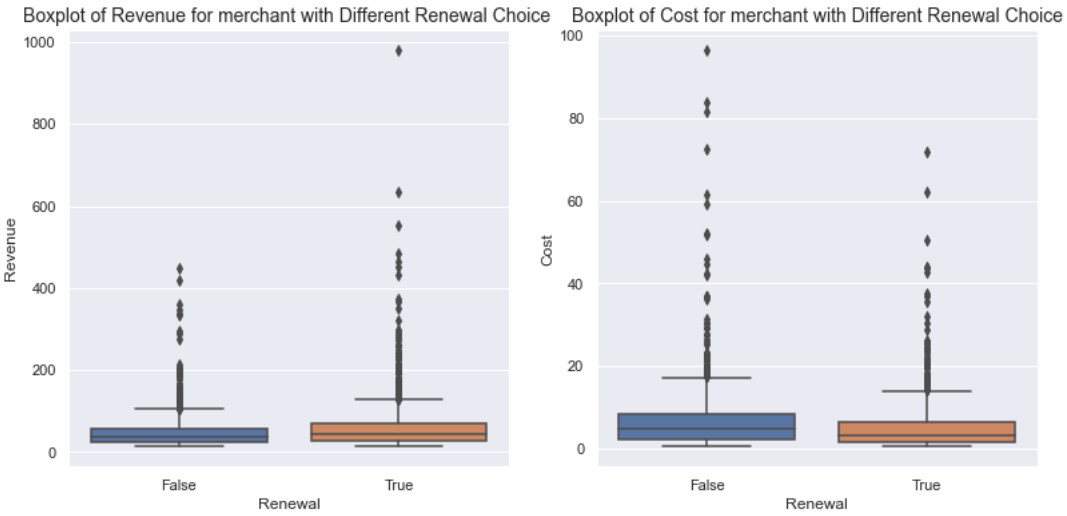
3.2.2 Revenue

However, boxplot is not directly applicable to analyse `Revenue` and `Cost` . As evidenced by the plot, the data is heavily **right-skewed by a few outliers**. The main body of the boxplot is condensed towards the bottom of the plot, making it challenging to carry out a meaningful analysis.

In [11]:

```
### Boxplot side by side
import matplotlib.pyplot as plt
import seaborn as sns
# Creation of figure with 2 axis
sns.set(style="ticks")
sns.set_style("darkgrid")
fig, ax = plt.subplots(1, 2, figsize=(13, 6))
# Creation of 1st axis
sns.boxplot(x="Renewal", y="Revenue", data=data_south, ax=ax[0])
ax[0].set_title("Boxplot of Revenue for merchant with Different Renewal Choice ", fontsize=14)
# Creation of 2nd axis
sns.boxplot(x="Renewal", y="Cost", data=data_south, ax=ax[1])
ax[1].set_title("Boxplot of Cost for merchant with Different Renewal Choice", fontsize=14)
# Close the empty Figure 2 created by seaborn.
plt.close(2)
```

executed in 106ms, finished 23:53:31 2023-02-13

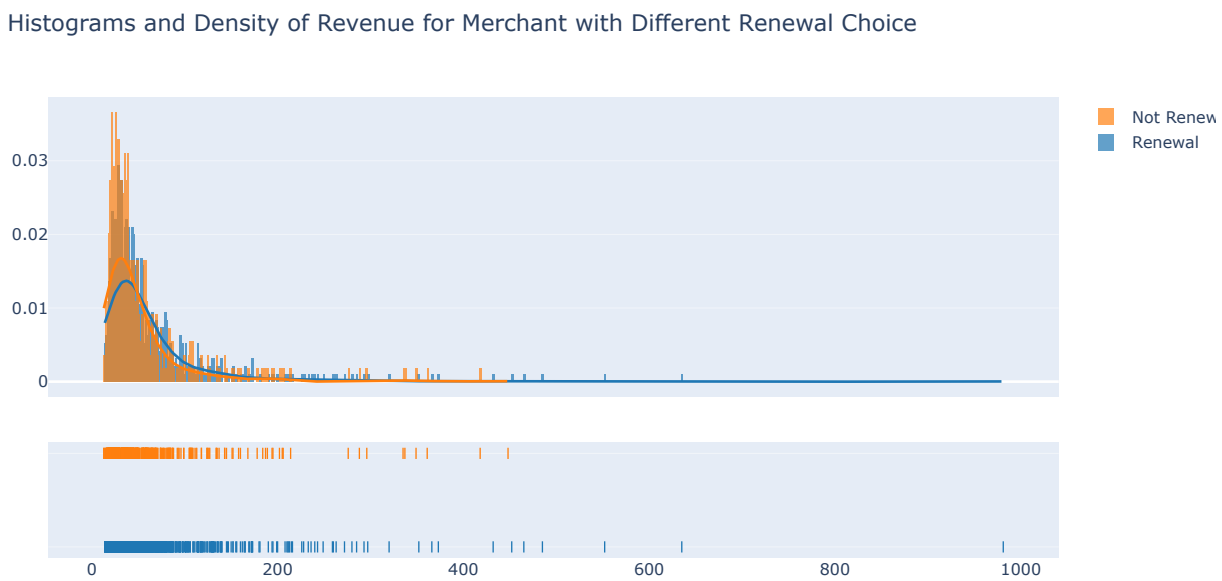


The visualization below aims to utilize the distribution of data for analysis. The plot displays histograms and density plots of Revenue for merchants with different contract renewal choices, with the lower rug plot showing the distribution of samples through individual vertical lines. **The data suggests that merchants with lower revenues are less likely to renew their contracts, while those with higher revenues tend to renew their contract**. However, this trend is not clearly defined and further analysis is needed to fully understand the relationship between Revenue and contract renewals.

In [12]:

```
### Histograms and Density plot of Revenue
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objs as go
fig = ff.create_distplot(
    [data_south.query("Renewal == True")['Revenue'],
     data_south.query("Renewal == False")['Revenue']],
    ["Renewal", "Not Renewal"])
fig.update_layout(bargap=0.1,
                  title_text = 'Histograms and Density of Revenue for Merchant with Different Renewal Choice')
fig
```

executed in 685ms, finished 23:53:32 2023-02-13



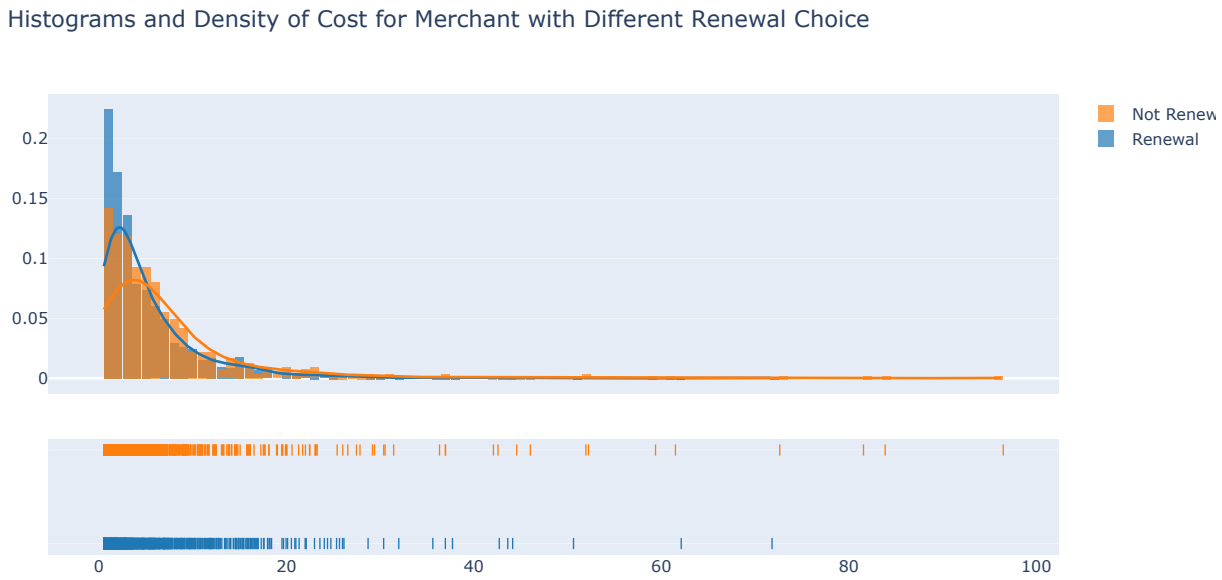
3.2.3 Cost

We can also analyze the `Cost` variable using a similar visualization. The results seem to indicate that merchants with higher rental cost are less likely to renew their contracts, while those with lower cost are more inclined to do so. However, this pattern is not significant again.

In [13]:

```
### Histograms and Density plot of Revenue
fig = ff.create_distplot(
    [data_south.query("Renewal == True")['Cost'],
     data_south.query("Renewal == False")['Cost']],
    ["Renewal", "Not Renewal"])
fig.update_layout(bargap=0.1)
fig.update_layout(bargap=0.1,
                  title_text = 'Histograms and Density of Cost for Merchant with Different Renewal Choice')
fig
```

executed in 52ms, finished 23:53:32 2023-02-13



3.2.4 Further discussion

When `Revenue` is plotted on the Y-axis and `Cost` on the X-axis, a scatter plot can be constructed and a linear model can be fit. The **slope of this linear model**

$$k = \frac{Revenue}{Cost} \tag{1}$$

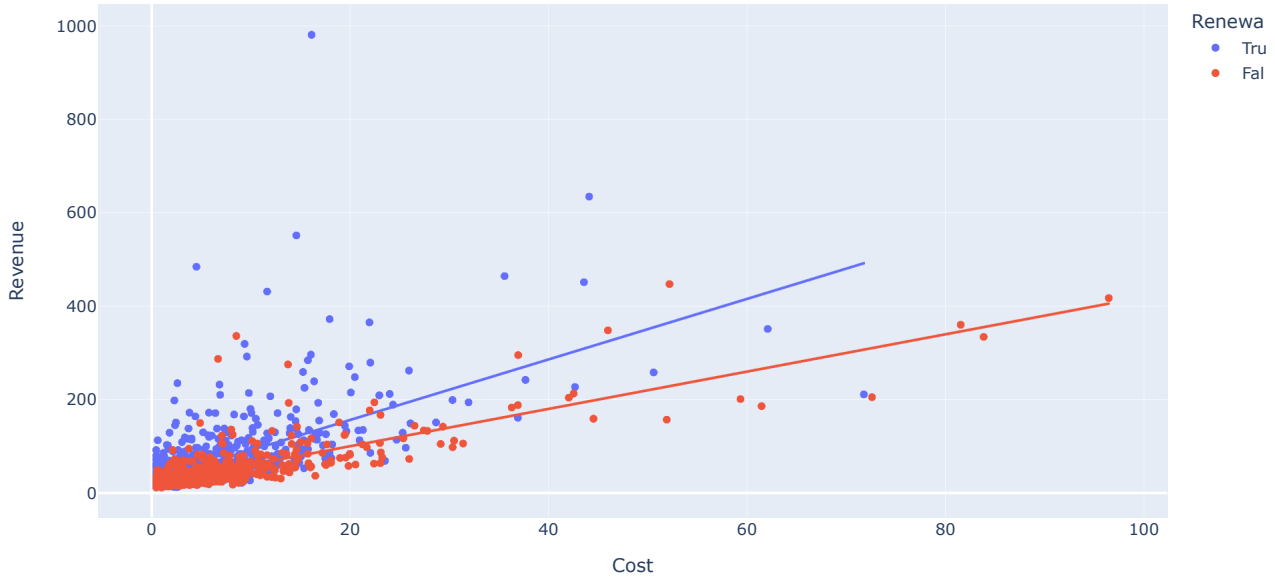
represents the profitability, which is the **revenue per unit rental cost**. An analysis of the data reveals that merchants who renew their contracts tend to exhibit strong profitability ($k = 6.476$), while merchants who do not renew their contracts often exhibit low profitability ($k = 3.990$). What a clear boundary!

It should be noted that the slopes of the lines in the figure may not appear to vary significantly due to the use of different scales for the X and Y axes. You may moving the cursor onto the line to obtain a clearer representation.

In [14]:

```
# Plot Revenue w.r.t. Cost
fig = px.scatter(data_south,
                 x = 'Cost',
                 y = 'Revenue',
                 color = 'Renewal',
                 trendline = "ols")
# fig.update_yaxes(scaleanchor="x", scaleratio=1)  # scale the axis
fig
```

executed in 509ms, finished 23:53:32 2023-02-13



3.3 Data preprocessing

The type of the data in the dataset is double, eliminating the need for additional preprocessing. The only required step is to **split the data into training and testing sets** to establish the KNN model and Naive Bayes Classifier.

3.3.1 Split the training and testing set

In [15]:

```
### Split the training and testing set
# Explanatory variables
x_south = data_south[['Registration_Duration', 'Revenue', 'Cost']]
# Response variable
y_south = data_south['Renewal']

# Split the dataset into training set (70%) and testing set (30%), the random_state is used to control randomness
x_south_train, x_south_test, y_south_train, y_south_test = train_test_split(
    x_south, y_south, test_size=0.3, random_state=1)
```

executed in 4ms, finished 23:53:32 2023-02-13

In [16]:

```
### check the shape
x_south_train.shape
```

executed in 2ms, finished 23:53:32 2023-02-13

Out[16]:

```
(1050, 3)
```

Typesetting math: 100%

Summary:

In the marginal variable analysis of the dataset `data_south`, we investigated the relationship between the explanatory variables `Registration_Duration`, `Revenue`, and `Cost` and the decision of the renewal contract. We found that:

- A positive correlation is established between `Registration_Duratio` n and `renewal`, indicating that merchants who have been registered for a longer period of time are more likely to renew their contract.
- The `Revenue` distribution is heavily right-skewed, and the histogram plot shows that merchants with higher income tend to renew their contracts, though the correlation is not significant.
- The `Cost` distribution is also skewed to the right, and the bar chart demonstrated that merchants with lower rental costs are more likely to renew their contracts.
- Additionally, we observed that merchants who make different renewal decisions also exhibit differences in their `Revenue per unit of Cost`.

Finally, we split the dataset `data_south` into a training set and a testing set for subsequenct analysis.

4 Model building

We will use the KNN model and the Naive Bayesian Classifier to model and predict whether the merchants will renew the contracts. The cross-validation method will be used to compare the two models.

4.1 KNN model

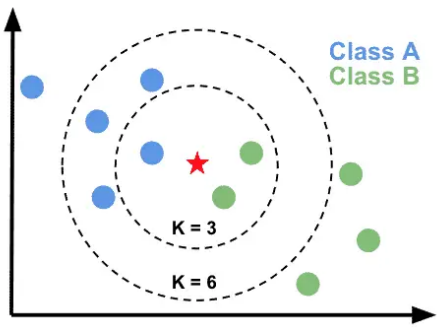
The **K-nearest neighbor (KNN) model** (https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) is a straightforward algorithm that is utilized for solving classification problems. The basic principle behind this model is to classify a new observation based on its proximity to other observations within the training data.

The **proximity** used is usually the Euclidean distance, which calculates the straight-line distance between two points in a multidimensional space. However, other proximity measures such as Manhattan distance, Minkowski distance, and cosine similarity can also be utilized:

- Euclidean distance: $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$,
- Manhattan distance: $\sum_{i=1}^{n}|x_i - y_i|$,
- Minkowski distance: $\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$,
- Cosine similarity: $\frac{\sum_{i=1}^{n}x_i y_i}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}}$,

  where x and y are two points or vectors in n-dimensional space, and x_i and y_i are their respective ith dimensions. The value of p in the Minkowski distance determines the type of Minkowski distance, where $p = 2$ corresponds to Euclidean distance and $p = 1$ corresponds to Manhattan distance.

The **value of K** in the KNN model determines the size of the neighborhood of observations that will be used to make a prediction. For instance, if K=3, then the three nearest neighbors to a new observation will be used to make a prediction. The prediction is the most frequent class among the neighbors. **In the case of a tie in even K, the textbook gives the solution of choosing a class at random.** In order to avoid ties in the voting, we only choose odd values for K (https://stats.stackexchange.com/questions/144718/how-does-scikit-learn-resolve-ties-in-the-knn-classification) in the following analyses.



As the textbook mentioned, the KNN model is a lazy learning algorithm, which means that it does not build a model during the training phase, but rather stores the training data. During the prediction phase, the KNN model calculates the distance between the new observation and all the training data, and selects the K nearest neighbors to make a prediction. Therefore, the **KNN model can be computationally expensive**, especially for large datasets.

In conclusion, the KNN model is a simple and efficient machine learning algorithm that can be used for classification problems. Careful consideration must be given to the choice of K to ensure optimal performance.

4.1.1 Find the optimal K

As mentioned above, the selection of K is crucial to the accuracy and performance of the model:

- a small value of K implies that the neighbourhood size is small and the model will be more susceptible to outliers and fluctuations in the data,
- a large value of K signifies a larger neighbourhood size and a more robust model to outliers and fluctuations, but it may not effectively capture the underlying structure of the data.

If we select a **very small K = 1**, then score of the model, which is the proportion of correct prediction made by the model, is **0.6378**:

In [17]:

```
# Create a KNN classifier object and set the number of neighbors to 1
knn_south = KNeighborsClassifier(n_neighbors=1)
# Train the model by the training set
knn_south.fit(x_south_train, y_south_train)
knn_south.score(x_south_test, y_south_test)
```

executed in 11ms, finished 23:53:32 2023-02-13

Out[17]:

```
0.6377777777777778
```

Then we try a very **large value K = 525**, which corresponds to half the total number of observations in the training data set, the score of the model is **0.6356**:

In [18]:

```
# Create a KNN classifier object and set the number of neighbors to 3
knn_south = KNeighborsClassifier(n_neighbors=525)
# Train the model by the training set
knn_south.fit(x_south_train, y_south_train)
knn_south.score(x_south_test, y_south_test)
```

executed in 36ms, finished 23:53:32 2023-02-13

Out[18]:

```
0.6355555555555555
```

As we will observe later, a **moderate value of K in the KNN algorithm can result in significant improvements in accuracy**. A well-selected value of K can balance the trade-off between overfitting and underfitting, leading to a more robust and accurate model.

4.1.1.1 Traverse all candidate parameters to find the optimal number of neighbors

We implements a function `**knn_model_evaluation**` to evaluate the performance of a KNN. The function takes two inputs, `x` and `y`, which are the explantory and response variables, respectively. The function also takes an optional parameter `range` which is a range of values for `k`, where `k` represents the number of nearest neighbors in the KNN model. By default, the range is set to `range(2, 30, 2)`, which means the function will evaluate the KNN model for `k` values from 2 to 30 with a step of 2 (only take odd k).

For each value of `k`, the function **creates a KNN model** and **uses 10-fold cross-validation** to evaluate the model. [Cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics))](https://en.wikipedia.org/wiki/Cross-validation_(statistics)) is a technique to evaluate the model's performance by dividing the dataset into 10 parts, using 9 parts to train the model, and 1 part to test the model. This process is **repeated 10 times**, and the average of the 10 results is used to evaluate the model's performance.

The function **calculates four metrics** to evaluate the KNN model: accuracy, precision, recall, and F1 score:

- Accuracy measures how often the model makes the correct prediction.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} \tag{2}$$

- Precision measures the proportion of positive predictions that are correct.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3}$$

- Recall measures the proportion of positive cases that the model identifies correctly.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4}$$

- F1 score is a balance between precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

The function stores the values of these metrics for each value of `k` and **aggregates the results** into a dataframe.

Finally, the function **plots a line graph to visualize the performance** of the KNN model for different values of `k`. The idea is to identify the optimal value of `k` that outstanding on all four evaluation metrics, as depicted in the graph.

In [19]:

```
### Define a funciton to evaluation KNN models
def knn_model_evaluation(x, y, range=range(2, 30, 2)):
    # Store the number of the neighbors
    ks = []
    # Store the accuracy
    accuracy_means = []
    # Store the precision
    precision_means = []
    # Store the recall
    recall_means = []
    # Store the F1
    f1_means = []

    for k in range:
        # Store the n_neighbors parameter
        ks.append(k)
        knnModel = KNeighborsClassifier(n_neighbors=k)
        # Calculate the accuracy of the 10-fold cross-validation
        accuracy_cvs = cross_val_score(
            knnModel,
            x, y, cv=10,
            scoring=make_scorer(accuracy_score)
        )
        # Store the average accuracy of 10-fold cross-validation
        accuracy_means.append(accuracy_cvs.mean())
        # Calculate the precision of the 10-fold cross-validation
        precision_cvs = cross_val_score(
            knnModel,
            x, y, cv=10,
            scoring=make_scorer(
                precision_score,
                pos_label=True # specify the positive label
            )
        )
        # Store the average precision of 10-fold cross-validation
        precision_means.append(precision_cvs.mean())
        # Calculate the recall of the 10-fold cross-validation
        recall_cvs = cross_val_score(
            knnModel,
            x, y, cv=10,
            scoring=make_scorer(
                recall_score,
                pos_label=True
            )
        )
        # Store the average recall of 10-fold cross-validation
        recall_means.append(recall_cvs.mean())
        # Calculate the f1 of the 10-fold cross-validation
        f1_cvs = cross_val_score(
            knnModel,
            x, y, cv=10,
            scoring=make_scorer(
                f1_score,
                pos_label=True
            )
        )
        # Store the average f1 of 10-fold cross-validation
        f1_means.append(f1_cvs.mean())

    # Aggregate the model scores
    scores = pandas.DataFrame({
        'k': ks,
        'precision': precision_means,
        'accuracy': accuracy_means,
        'recall': recall_means,
        'f1': f1_means
    })

    # Plot the line graph of different scores
    plt.figure(figsize=(8, 6))
    scores.plot(
        x='k',
        y=['accuracy', 'precision', 'recall', 'f1']
    )
```

executed in 5ms, finished 23:53:32 2023-02-13

Typesetting math: 100%

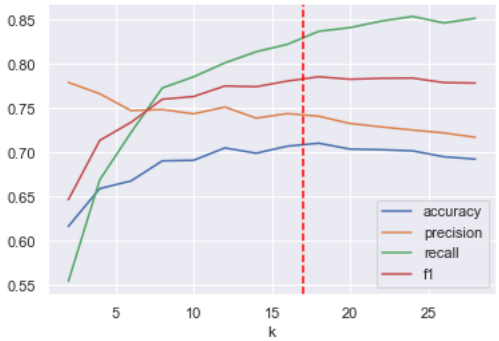
In [20]:

```
### Evaluate the odd k from 2 to 29
knn_model_evaluation(x_south, y_south, range=range(2,30,2))

# we pick k=17 as the optimal number of neighbors
plt.axvline(x = 17, color = 'red', linestyle = '--')
```

executed in 2.44s, finished 23:53:35 2023-02-13

Out[20]:

```
<matplotlib.lines.Line2D at 0x7ff0e13f1790>

<Figure size 576x432 with 0 Axes>
```



As observed from the plot above:

- `Accuracy` increases for `k<18`, but gradually decreases for `k>18`.
- `Precision` decreases with the increase of `k`, particularly when `k>17`.
- `Recall`, increases with the increase of `k` and experiences a rapid rise when `k<17`, with a smaller increment when `k>17`.
- `F1` continues to rise with the increase of `k`, with a faster rise when `k<9` and a smaller increment when `k>9`.

Based on these observations, it can be concluded that the model performs optimally on all four evaluation metrics when `k=17`, as indicated by the red dashed line in the plot. Therefore, **`k=17` is selected as the optimal value**.

Here, we build the KNN model with the optimal `k` based on the training set:

In [21]:

```
# Create a KNN classifier object with the optimal n_neighbors=17
knn_south_17 = KNeighborsClassifier(n_neighbors=17)
# Train the model by the training set
knn_south_17.fit(x_south_train, y_south_train)
```

executed in 5ms, finished 23:53:35 2023-02-13

Out[21]:

```
KNeighborsClassifier(n_neighbors=17)
```

4.2 Gussian native bayesian model

The **Naive Bayes Classifier** is a machine learning algorithm that is used for **classification** purposes. It is a probabilistic model that makes predictions based on the probability of an event occurring given certain conditions.

The algorithm is called "**Naive**" because it **assumes that all features are independent of one another**, which is not always the case in real-world situations. However, the Naive Bayes Classifier still performs well in many applications including text classification and spam filtering.

The basic idea behind the algorithm is to calculate the probabilities of a given item belonging to each class and then selecting the class with the highest probability. The probabilities are calculated using **Bayes' theorem**:

$$P(C_i|X) = \frac{P(X|C_i) * P(C_i)}{P(X)} \propto P(X|C_i) * P(C_i) \tag{6}$$

where:

- $P(C_i|X)$ is the posterior probability of class C_i given predictor X.
- $P(X|C_i)$ is the likelihood which is the probability of predictor X given class C_i.
- $P(C_i)$ is the prior probability of class C_i.
- $P(X)$ is the marginal probability of predictor X, often referred to as evidence (https://en.wikipedia.org/wiki/Marginal_likelihood).

In the calculation of probabilities, it is possible to encounter instances where the count of a feature occurring in a class is zero. This would result in a **zero probability**, which would in turn affect the accuracy of the classifier. The **Laplace transformation** is used to add a small constant value to the count to the numerator and denominator, thereby avoiding the issue of zero probabilities.

The **likelihood** of predictor x_i given class C_i can be calculated as:

$$P(x_i|C_j) = \frac{n_{ij} + 1}{n_j + N} \tag{7}$$

where

- $P(x_i|C_j)$ is the probability of feature x_i occurring given class C_j
- n_{ij} is the number of times feature x_i occurs in class C_j
- n_j is the total number of times features occur in class C_j
- N is the number of features

The **prior** of class C_j can be calculated as:

$$P(C_j) = \frac{n_j + 1}{n + K} \tag{8}$$

where

- $P(C_j)$ is the prior probability of class C_j
- n_j is the number of samples in class C_j
- n is the total number of samples
- K is the number of classes

The Naive Bayes Classifier **allows different types of probability distributions**, depending on the nature of the data:

- ***Gaussian Naive Bayes Classifier*** assumes that the ***continuous*** features follow a normal distribution. That is to say, for each feature, the classifier **estimates the mean and variance** to obtain the corresponding normal distribution. For each new observation, likelihood can be obtained from the normal distribution.
- ***Multinomial Naive Bayes Classifier*** is used for ***discrete*** data, such as text classification. It calculates the probability of each word in the text belonging to a particular class.
- ***Bernoulli Naive Bayes Classifier*** is used for ***discrete*** data. It is similar to the Multinomial Naive Bayes, but it ***only considers whether a word appears*** in the text or not, rather than the frequency of the word.

In this tutorial, we will use the Gaussian Naive Bayes Classifier to model the contract renewal problem, since the explantory variables in the dataset are continous.

Unlike in KNN models, where we must determine the value of `k` that greatly affects the accuracy of the model. In Naive Bayes Classifier, we **hardly need to determine any parameters** besides the probability distribution to be used. This reduces the threshold for using the model and the risk of overfitting. In the code below, we establish a **Gaussian Naive Bayes Classifier**:

In [22]:

```
# Create a Gussian native bayesian classifier object
gaussianNB = GaussianNB()
# Train the model by the whole dataset
gaussianNB.fit(x_south_train, y_south_train)
```

executed in 4ms, finished 23:53:35 2023-02-13

Out[22]:

```
GaussianNB()
```

5 Model comparison

5.1 Confusion matrices

We first compare models in terms of the **confusion matrices** obtained on the training data set.

The confusion matrix is a table used to evaluate the performance of a classifier. It shows the number or the proportion of correct and incorrect predictions made by the classifier for each class. The matrix is usually presented as a 2x2 table for binary classification problems, where one class represents positive (1) class and the other class represents negative (0) class. The **rows** in the table represent the **actual** class, while the **columns** represent the **predicted** class.

The cells in the table represent the number or the proportion of instances that belong to a specific class and are correctly or incorrectly predicted by the classifier. For example, a confusion matrix for a binary classification problem would look like this:

	Predicted Class 0	Predicted Class 1	Total
Actual Class 0	True Negative (TN)	False Positive (FP)	All Positive (AP)
Actual Class 1	False Negative (FN)	True Positive (TP)	All Negative (AN)

where

- `True Negative (TN)` represents the number of instances that are **actually negative** and are correctly **predicted as negative** by the classifier.
- `True Positive (TP)` represents the number of instances that are **actually positive** and are correctly **predicted as positive** by the classifier.
- `False Positive (FP)` , also known as the **Type 1 error**, represents the number of instances that are **actually negative** but are falsely **predicted as positive** by the classifier.
- `False Negative (FN)` , also known as the **Type 2 error**, represents the number of instances that are **actually positive** but are falsely **predicted as negative** by the classifier.
- `All Positive` represents the number of **positive observations**.
- `All Negative` represents the number of **negative observations**.

The cells in the confusion matrix can be used to calculate various performance metrics such as **accuracy**, **precision**, **recall**, and **F1 score**.

Dealing with unbalanced classes: normalizing the confusion matrix by row

When the dataset is very **imbalanced** (the number of positive and negative samples is very different), the confusion matrix obtained by direct counting is not very meaningful. In this case, **normalizing the confusion matrix by row** (dividing each cell by the sum of its corresponding row) will give you a better idea of the classifier's performance. Normalization by row shows the proportion of positive and negative samples predicted correctly and incorrectly by the classifier, regardless of the total number of positive and negative samples:

	Predicted Class 0	Predicted Class 1	Total
Actual Class 0	True Negative Rate	False Positive Rate	1
Actual Class 1	False Negative Rate	True Positive Rate	1

For example, a Naive Bayes Classifier is used to predict the win or loss of a horse race with 10 horses, but the classifier might predict all the horses as lost. Here's the confusion matrix:

	Predicted Loss	Predicted Win
Actual Loss	9	0
Actual Win	1	0

It looks like we have correctly predicted the outcomes of 9 horses and only incorrectly predicted 1 horse, but this classifier is obviously useless.

And the normalized confusion matrix:

	Predicted Loss	Predicted Win
Actual Loss	100%	0
Actual Win	100%	0

From the normalized confusion matrix, the classifier correctly predicted 100% of horses that lost the race, and incorrectly predicted 100% of horses that won the race. In this way, by eliminating the effect of the difference in the number of samples in the class, we have shown the shortcomings of the classifier. **In the following analysis. We will use this technique.**

In [23]:

```
# Define a function to calculate the confusion matrix
from sklearn.metrics import confusion_matrix # import confusion matrix module
def calculate_confusion_matrix(y_true, y_pred, labels):
    confu_mat = pandas.DataFrame(confusion_matrix(y_true, y_pred,
                                                  normalize='true',  # calculate the proportion
                                                  labels=labels)) * 100  # pos label = True
    return round(confu_mat, 2).astype(str).apply(lambda x: x + '%')  # formatting the output
```

executed in 2ms, finished 23:53:35 2023-02-13

In [24]:

```
# Confusion matrix of KNN model with k=17
calculate_confusion_matrix(y_south_test, knn_south_17.predict(x_south_test), labels=[False, True])
```

executed in 13ms, finished 23:53:35 2023-02-13

Out[24]:

	0	1
0	40.85%	59.15%
1	12.94%	87.06%

For customers who want to renew the contract, the KNN model successfully predicts 87.06% of the samples, and predicts 12.94% of the samples as unwilling to renew. For the customers who do not want to renew, the knn model successfully predicts 40.85% of the samples, and mistakenly forecast that 59.15% of the customers want to renew. Overall, **the KNN model performs well in predicting the customers that want to renew, reaching 87%, and perform bad in predicting the customers that do not want to renew, only 40%**.

In [25]:

```
# Confusion matrix of Gussian native bayesian classifier
calculate_confusion_matrix(y_south_test, gaussianNB.predict(x_south_test), labels=[False, True])
```

executed in 5ms, finished 23:53:35 2023-02-13

Out[25]:

	0	1
0	37.8%	62.2%
1	15.38%	84.62%

From the confusion matrix above, we can see that the Gaussian Bayes classifier successfully distinguished 84.62% of the customers who wanted to renew their contract, which is lower than the KNN model's 87.06%. For customers who do not want to renew their contracts, the Gaussian Bayesian classifier successfully predicts 37.80% of the samples, which is also lower than the knn model's 40.85%. In other words, **the Gaussian Bayes classifier is slightly lower than the KNN model in the ability to judge both positive and negative samples**. It seems that **the KNN model with careful parameter tuning outperforms the Gaussian Bayes classifier on the testing set**.

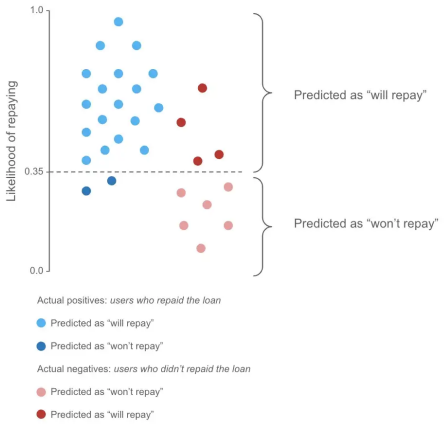
5.2 ROC curve and AUC

A **ROC curve (Receiver Operating Characteristic curve)** is a graphical representation of the performance of a binary classifier. The ROC curve is created by plotting the `True Positive Rate` against the `False Positive Rate` at various **threshold values**.

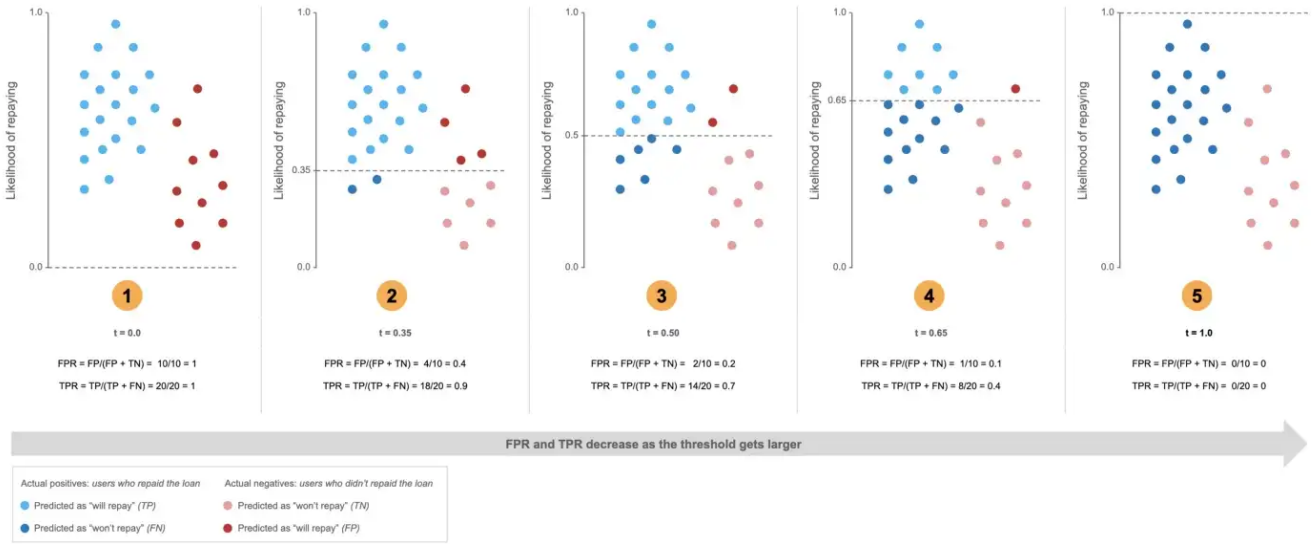
The **area under the ROC curve (AUC)** is a single number that summarizes the overall performance of the classifier by measuring the extent to which the classifier can distinguish between positive and negative samples. The AUC ranges from **0 (indicating a poor classifier)**, to **1 (indicating a perfect classifier)**. A classifier with an AUC of **0.5 is equivalent to random guessing**, whereas a classifier with an AUC of 1 can perfectly distinguish between positive and negative samples.

To plot a ROC curve, the classifier must first be trained and tested on a dataset. Then the `True Positive Rate` and `False Positive Rate` are calculated at various threshold values*. The FPR and TPR are then plotted against each other to generate the ROC curve. Here is an example on the likelihood of repaying a loan (https://towardsdatascience.com/understanding-the-roc-curve-in-three-visual-steps-795b1399481c):

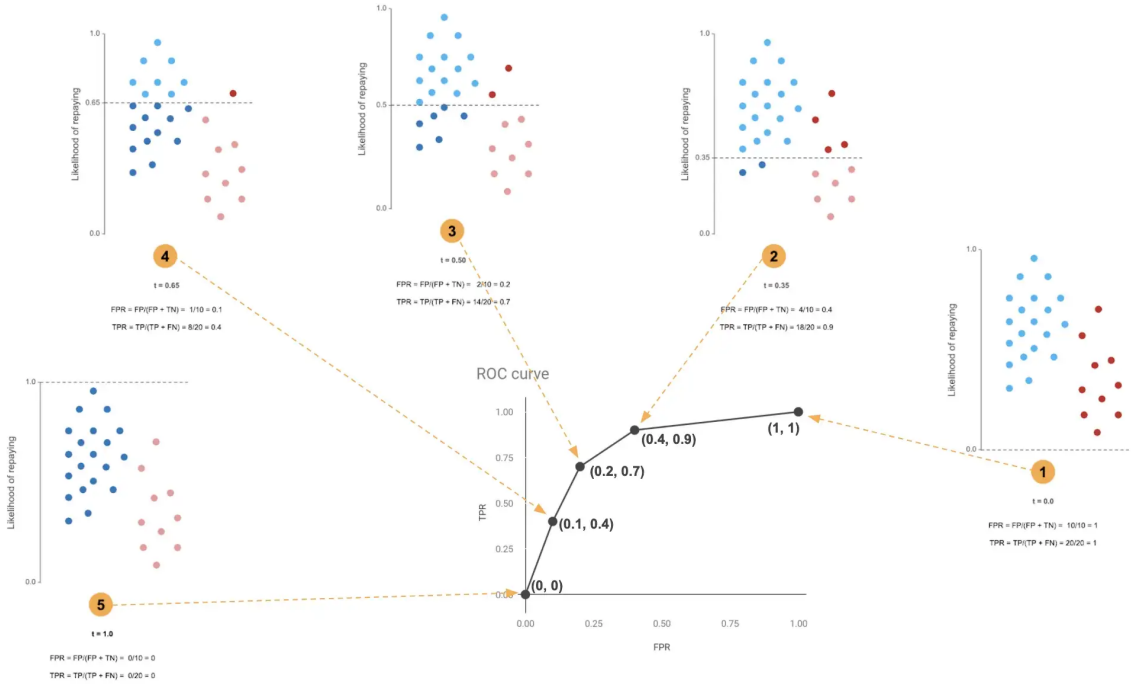
Step 1: Getting classification model predictions
The likelihood usually range between 0 and 1. The higher the value, the more likely the person is to repay a loan.



Step 2: Calculate the True Positive Rate and False Positive Rate
We need to calculate the True Positive Rate and False Positive Rate with respect to different threshold values. Overal l, we can see this is a trade-off. As we increase our threshold, we'll be better at classifying negatives, but this is at the expense of miss-classifying more positives



Step 3: Plot the the TPR and FPR for every cut-off
For each threshold, we plot the FPR value in the x-axis and the TPR value in the y-axis. We then join the dots with a line. That's it!
The area covered below the line is called "AUC". This is used to evaluate the performance of a classification model. T he higher the AUC, the better the model is at distinguishing between classes.



In conclusion, the ROC curve and AUC are important tools for evaluating the performance of binary classifiers. The ROC curve allows us to visualize the trade-off between the true positive rate and false positive rate, and the AUC summarizes the overall performance of the classifier by measuring the extent to which it can distinguish between positive and negative samples.

**Some may be confused by the ROC curve obtained from models like KNN models and Naive Bayes Classifier. Since there is no threshold values in these models. Actually, the threshold values in the context of KNN can be the number of neighbours that correctly classify an instance (the threshold), divided by the total number of neighbours used (the k parameter).*
E.g., you have a KNN model f with k=5, and for data instance x, the score = 4 / 5 if 4 of its 5 closest neighbours belong to the same class as x. If for data instance y you have score = 3 / 5, then both x and y would be assigned the same class, but x should rank higher than y because the model is more confident about it (4/5 > 3/5). In fact, his is what's implemented in the predict_proba method of the sklearn's KNN classifier.
For more information, you may refer to:
ROC curve from models using k-nearest neighbor algorithms (https://community.rapidminer.com/discussion/19497/roc-curve-from-models-using-k-nearest-neighbor-algorithms)
Is it sensible to use the ROC curve with an KNN model? And if so why? (https://datascience.stackexchange.com/questions/117075/is-it-sensible-to-use-the-roc-curve-with-an-knn-model-and-if-so-why)

In [26]:

```
### Define a funtion to plot the ROC curve and AUC
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

def plot_roc_curve(model1, model2, x_test, y_test, model1_name, model2_name):
    # Fit the first model
    model1.fit(x_test, y_test)
    # Predict the probabilities of positive class
    y_pred_proba_model1 = model1.predict_proba(x_test)[:, 1]
    # Calculate the false positive rate (FPR), true positive rate (TPR), and thresholds
    fpr_model1, tpr_model1, thresholds_model1 = roc_curve(y_test, y_pred_proba_model1)
    # Calculate the AUC for the first model
    auc_model1 = roc_auc_score(y_test, y_pred_proba_model1)

    # Fit the second model
    model2.fit(x_test, y_test)
    # Predict the probabilities of positive class
    y_pred_proba_model2 = model2.predict_proba(x_test)[:, 1]
    # Calculate the false positive rate (FPR), true positive rate (TPR), and thresholds
    fpr_model2, tpr_model2, thresholds_model2 = roc_curve(y_test, y_pred_proba_model2)
    # Calculate the AUC for the second model
    auc_model2 = roc_auc_score(y_test, y_pred_proba_model2)

    plt.figure(figsize=(8, 6))
    # Plot the ROC curve for the first model
    plt.plot(fpr_model1, tpr_model1, label='{} ROC curve (AUC = {:.3f})'.format(model1_name, auc_model1))
    # Plot the ROC curve for the second model
    plt.plot(fpr_model2, tpr_model2, label='{} ROC curve (AUC = {:.3f})'.format(model2_name, auc_model2))
    plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve for {} and {}'.format(model1_name, model2_name))
    plt.legend()
    plt.show()
```

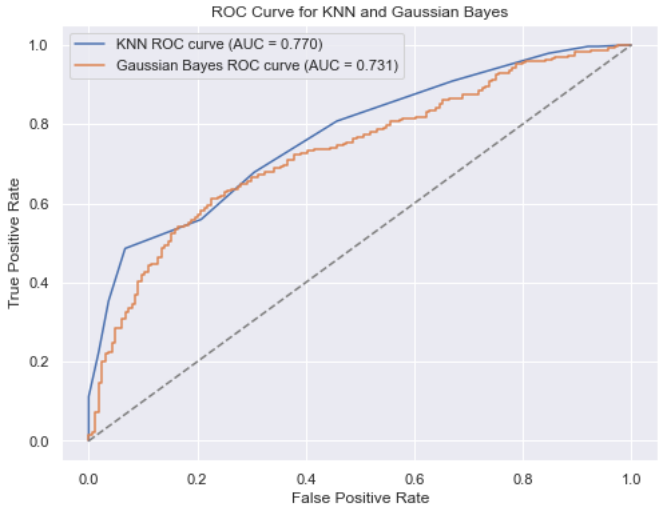
executed in 5ms, finished 23:53:35 2023-02-13

Typesetting math: 100%

In [27]:

```
# Plot the ROC curve for KNN and Naive Bayes
plot_roc_curve(knn_south_17, gaussianNB, x_south_test, y_south_test, 'KNN', 'Gaussian Bayes')
```

executed in 86ms, finished 23:53:35 2023-02-13



In the **ROC curve** above, the horizontal axis is the False Positive Rate and the vertical axis is the True Positive Rate. the **dashed gray line** from the bottom left to the top right of the figure is the **random guessing line**. It can be seen that the ROC curves of KNN model and Gaussian Bayes classifier are better than random guessing line. Since both ROC curves are computed through samples, they are not smooth curves. With the increase of FPR, the ROC curve of the KNN model grows very fast, which indicates that the **KNN model can distinguish more positive samples while judging fewer negative samples as positive samples**. This is in line with the results of our confusion matrix analysis (the KNN model has a strong ability to predict the merchants that want to renew the contract). **The ROC curve of the KNN model is higher than that of the Gaussian Bayesian classifier in most cases**, with only a little crossover around FPR=0.2. In terms of AUC value, the **AUC of KNN model is 0.770**, which is larger than the **AUC of Gaussian Bayesian classifier 0.731**.

In short, the KNN model is better than the Gaussian Bayesian classifier in the comparison of confusion matrix and ROC curve. Without considering the computational complexity, **the KNN model (k=17) should be selected as the classification model to predict whether the merchant will renew the contract**.

6 Predict the response in North China

In the following, we use two classification models to predict the renewal of merchants in South China.

In [28]:

```
### Predict the response in North China by KNN model and Gaussian Bayesian classifier
data_north['Predict_response_knn'] = knn_south_17.predict(
    data_north[['Registration_Duration', 'Revenue', 'Cost']]
)
data_north['Predict_response_gaussianNB'] = gaussianNB.predict(
    data_north[['Registration_Duration', 'Revenue', 'Cost']]
)
data_north.head()
```

executed in 13ms, finished 23:53:35 2023-02-13

Out[28]:

	ID	Registration_Duration	Revenue	Cost	Predict_response_knn	Predict_response_gaussianNB
0	20001	29	46	8.33316	False	True
1	20002	26	32	7.19040	False	True
2	20003	59	172	10.11150	True	True
3	20004	22	24	0.95979	True	True
4	20005	56	87	17.97600	True	True

In [29]:

```
### Contingency tables of the prediction result
pandas.crosstab(index=data_north['Predict_response_knn'],
                                columns=data_north['Predict_response_gaussianNB'],
                margins='all')
```

executed in 16ms, finished 23:53:35 2023-02-13

Out[29]:

Predict_response_gaussianNB	False	True	All
Predict_response_knn			
False	4	54	58
True	20	247	267
All	24	301	325

The above is the contingency table of Gaussian Bayes Classifier and KNN model. It can be seen that among a total of 325 samples, the knn model judged 267 samples as renewal, and 58 samples as not renewal. However, the Gaussian Bayes classifier judged 301 samples as renewal and 24 as not renewal. **Both models reached consensus on 247 merchants that renewed, while only 4 merchants were simultaneously considered not to renew.**

7 Summary report

7.1 Context

In this study, we aim to help a store leasing company to predict the likelihood of merchants renewing their contracts in North China. We use both KNN models and Naive Bayes Classifiers and base our predictions on key factors such as `Duration of registration`, `Revenue`, and `Rental cost`. The historical data from South China is used to develop the model since the circumstances in both regions are similar. This information will aid the company's business department in making informed investment decisions.

7.2 Objectives

The object of the study is to develop a predictive model based on merchants' data to predict the probability of merchants in North China renewing their contracts.

7.3 Organisation of the data

The study uses two datasets: `data_south` and `data_north`. The `data_south` serves as the training data and has 1500 samples with 5 variables `ID`, `Registration_Duration`, `Revenue`, `Cost`, and `Renewal`. The `data_north` has 325 samples and 4 variables, with the `Renewal` variable missing and to be predicted.

7.4 Exploratory data analysis

In the marginal variable analysis of the dataset `data_south`, we investigated the relationship between the explanatory variables `Registration_Duration`, `Revenue`, and `Cost` and the decision of the renewal contract. We found that:

- A positive correlation is established between `Registration_Duratio` n and `renewal`, indicating that merchants who have been registered for a longer period of time are more likely to renew their contract.
- The `Revenue` distribution is heavily right-skewed, and the histogram plot shows that merchants with higher income tend to renew their contracts, though the correlation is not significant.
- The `Cost` distribution is also skewed to the right, and the bar chart demonstrated that merchants with lower rental costs are more likely to renew their contracts.
- Additionally, we observed that merchants who make different renewal decisions also exhibit differences in their `Revenue per unit of Cost`.

Finally, we split the dataset `data_south` into a training set and a testing set for subsequenct analysis.

7.5 Model specification

The objective of our analysis is to propose a predictive classification model that divides merchants into those who are willing to renew the contract and those who are not, and the features of the data set are **continuous** variables. Therefore, we consider using **KNN model** and **Gaussian Bayes classifier**.

7.6 Model comparison

We used the **confusion matrix** and **ROC curve** to compare the models. We obtained classification predictions for both models on the testing set. Then, a normalized confusion matrix was used to compare the classification errors of both models. And what we found is, the KNN model performs well in predicting the merchants that want to renew than in predicting the merchants that do not want to renew. In addition, **the Gaussian Bayes classifier is lower than the KNN model in judging both positive and negative samples**. Then, we compared the ROC curves and AUC values of the two models. The result is that **the ROC curve of the KNN model is almost always above the Gaussian Bayes classifier**, and **the AUC value of the KNN model is greater** than that of the Gaussian Bayes Classifier. In short, the KNN model is better than the Gaussian Bayesian classifier in the comparison of confusion matrix and ROC curve. Without considering the computational complexity, **the KNN model (k=17) should be selected as the classification model** to predict whether the merchant will renew the contract.

7.7 Model interpretation

Since KNN model is a lazy model, we do not need to interpret the parameters of the model. In the analysis, we find that the KNN model has better prediction effect on the merchants willing to renew the contract, while the merchants unwilling to renew the contract have worse prediction effect. We may guess that the merchants willing to renew the contract are **close to each other on several features to facilitate prediction**. Merchants unwilling to renew contracts may be **scattered in the feature space, making them difficult to predict**. That is to say, merchants who do not want to renew the contract may give up the contract renewal **due to reasons other than the dataset included**, which requires the enterprise to collect further data to prove.