Section 3: Linear Models

- in this section, we will look at
  - Linear regression model
  - Evaluating the linear regression model
  - Logistic regression model
  - Evaluating the logistic regression model

```
as pd
odel_selection import train_test_split
inear_model import LinearRegression
 as sns
line
lib.pyplot as plt
```

```
_csv('Datasets/Weather.csv')
```

```
n\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3049: DtypeWarning: Columns (7,8,18,25) ha
ecify dtype option on import or set low_memory=False.
y=interactivity, compiler=compiler, result=result)
```

te	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	YR	...	FB	FTI	ITH	PGT	TSHDSBRSGF	SD3
-1	1.016	NaN	25.555556	22.222222	23.888889	0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN
-2	0	NaN	28.888889	21.666667	25.555556	0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN
-3	2.54	NaN	26.111111	22.222222	24.444444	0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN
-4	2.54	NaN	26.666667	22.222222	24.444444	0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN
-5	0	NaN	26.666667	21.666667	24.444444	0	NaN	42	...	NaN	NaN	NaN	NaN	NaN	NaN

ns

In [5]:
```
final_data.head()
```
Out[5]:

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	PoorWeather	YR	...	FB	FTI	ITH	PGT	TSHDSBRSGF	SD3	RHX	RHN
0	10001	1942-7-1	1.016	0.0	25.555556	22.222222	23.888889	0	0	42	...	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
1	10001	1942-7-2	0	0.0	28.888889	21.666667	25.555556	0	0	42	...	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
2	10001	1942-7-3	2.54	0.0	26.111111	22.222222	24.444444	0	0	42	...	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
3	10001	1942-7-4	2.54	0.0	26.666667	22.222222	24.444444	0	0	42	...	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0
4	10001	1942-7-5	0	0.0	26.666667	21.666667	24.444444	0	0	42	...	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0

5 rows × 31 columns

```
ả sử chúng ta xây dựng mô hình cho cột MinTemp và cột MaxTemp
i toán predicting maximum temperature
final_data['MaxTemp'].values.reshape(-1,1)
final_data['MinTemp'].values.reshape(-1,1)
```

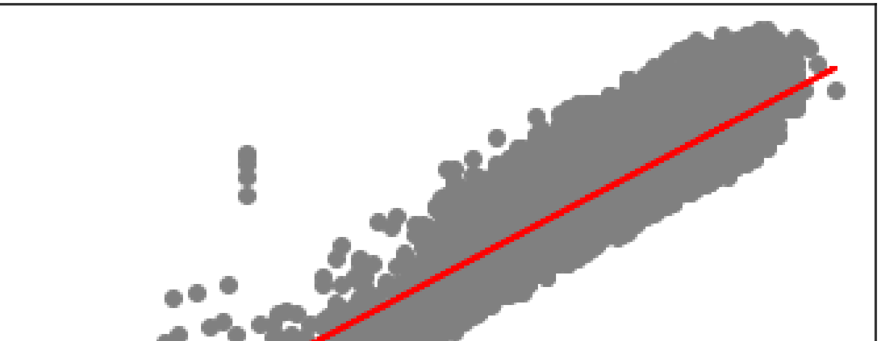
```
ia tập train, test
ain, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
i mô hình linear regression, học tập tập train
eg = LinearRegression()
eg.fit(X_train, y_train)
```

```
arRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
ed = linReg.predict(X_test)
```

```
 hình hóa với matplotlib
scatter([X_test], [y_test],  color='gray')
plot(X_test, y_pred, color='red', linewidth=2)
show()
```



Evaluating the linear regression model

- Gọi y là giá trị đúng, $\hat{y}$ là giá trị dự đoán của mô hình.
- Ta có n điểm dữ liệu.
- Mean Absolute Error (MAE): MAE = $\frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$
- Mean Square Error (MSE): MSE = $\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$
- Root Mean Square Error (RMSE): RMSE = $\sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
  - The most popular of the three

evaluate model performance

In [14]:
```
# calculate MAE, MSE, RMSE
from sklearn import metrics
import numpy as np

print("MAE:", metrics.mean_absolute_error(y_test, y_pred))
print("MSE:", metrics.mean_squared_error(y_test, y_pred))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 3.19932917837853
MSE: 17.631568097568447
RMSE: 4.198996082109204
```

In [ ]:

Logistic Regression

In [32]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
%matplotlib inline
import matplotlib.pyplot as plt
```

In [33]:
```
data = pd.read_csv('Datasets/Sales-Win-Loss.csv')
```

In [34]:
```
data.head()
```

Out[34]:

	Opportunity Number	Supplies Subgroup	Supplies Group	Region	Route To Market	Elapsed Days In Sales Stage	Opportunity Result	Sales Stage Change Count	Total Days Identified Through Closing	Total Days Identified Through Qualified	Opportunity Amount USD	Client Size By Revenue	Client Size By Employee Count	Revenue From Client Past Two Years	Compe
0	1641984	Accessories	Car	Northwest	Fields Sales	76	Won	13	104	101	0	5	5	0	Unk
1	1658010	Accessories	Car	Pacific	Reseller	63	Loss	2	163	163	0	3	5	0	Unk
2	1674737	Motorcycle Parts	Non-auto	Pacific	Reseller	24	Won	7	82	82	7750	1	1	0	Unk
3	1675224	Shelters	Non-auto	Midwest	Reseller	16	Loss	5	124	124	0	1	1	0	K
4	1689785	Accessories	Car	Pacific	Reseller	69	Loss	11	91	13	69756	1	1	0	Unk

In [36]:
```
# Assume that we predict "Opportunity Result"
lb = preprocessing.LabelBinarizer()
data['result'] = lb.fit_transform(data['Opportunity Result'])
```

In [37]:
```
# Convert categorical features into numerical format
le = preprocessing.LabelEncoder()

data['Supplies_Subgroup_Encoded'] = le.fit_transform(data['Supplies Subgroup'])
data['Supplies_Group_Encoded'] = le.fit_transform(data['Supplies Group'])
data['Region_Encoded'] = le.fit_transform(data['Region'])
data['Route_To_Market_Encoded'] = le.fit_transform(data['Route To Market'])
data['Competitor_Type_Encoded'] = le.fit_transform(data['Competitor Type'])
```

In [38]:
```
final_data = data.drop(['Supplies Subgroup', 'Supplies Group', 'Region', 'Route To Market', \
                        'Competitor Type', 'Opportunity Result'], axis=1)
```

In [39]:
```
final_data.head()
```

Out[39]:

	Opportunity Number	Elapsed Days In Sales Stage	Sales Stage Change Count	Total Days Identified Through Closing	Total Days Identified Through Qualified	Opportunity Amount USD	Client Size By Revenue	Client Size By Employee Count	Revenue From Client Past Two Years	Ratio Days Identified To Total Days	Ratio Days Validated To Total Days	Ratio Days Qualified To Total Days	Deal Size Category	result	Supplies_Subg
0	1641984	76	13	104	101	0	5	5	0	0.69636	0.113985	0.154215	1	1	
1	1658010	63	2	163	163	0	3	5	0	0.00000	1.000000	0.000000	1	0	
2	1674737	24	7	82	82	7750	1	1	0	1.00000	0.000000	0.000000	1	1	
3	1675224	16	5	124	124	0	1	1	0	1.00000	0.000000	0.000000	1	0	
4	1689785	69	11	91	13	69756	1	1	0	0.00000	0.141125	0.000000	4	0	

In [40]:
```
# Apply scaling feature values
continuous_features = ['Opportunity Number', 'Elapsed Days In Sales Stage', 'Sales Stage Change Count',
                       'Total Days Identified Through Closing', 'Total Days Identified Through Qualified',
                       'Opportunity Amount USD', 'Revenue From Client Past Two Years', 'Ratio Days Identified To Total Days',
                       'Ratio Days Validated To Total Days', 'Ratio Days Qualified To Total Days' ]
final_data[continuous_features] = MinMaxScaler().fit_transform(final_data[continuous_features])
```

```
C:\Users\platon\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:334: DataConversionWarning: Data with input dtype
int64, float64 were all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [41]:
```
final_data.head()
```

Out[41]:

	Opportunity Number	Elapsed Days In Sales Stage	Sales Stage Change Count	Total Days Identified Through Closing	Total Days Identified Through Qualified	Opportunity Amount USD	Client Size By Revenue	Client Size By Employee Count	Revenue From Client Past Two Years	Ratio Days Identified To Total Days	Ratio Days Validated To Total Days	Ratio Days Qualified To Total Days	Deal Size Category	result	Supplies_S
0	0.000000	0.361905	0.545455	0.500000	0.485577	0.000000	5	5	0.0	0.69636	0.113985	0.154215	1	1	
1	0.001896	0.300000	0.045455	0.783654	0.783654	0.000000	3	5	0.0	0.00000	1.000000	0.000000	1	0	
2	0.003875	0.114286	0.272727	0.394231	0.394231	0.007750	1	1	0.0	1.00000	0.000000	0.000000	1	1	
3	0.003933	0.076190	0.181818	0.596154	0.596154	0.000000	1	1	0.0	1.00000	0.000000	0.000000	1	0	
4	0.005655	0.328571	0.454545	0.437500	0.062500	0.069756	1	1	0.0	0.00000	0.141125	0.000000	4	0	

```
In [42]: # prepare data
         y = final_data["result"]
         X = final_data.loc[:, final_data.columns != "result"]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [43]: # build the model without removing any outliers
         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)
```

```
C:\Users\platon\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be cha
nged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
Out[43]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```

```
In [46]: y_pred = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.4f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.8357
```

```
In [45]: pd.crosstab(y_test, y_pred, rownames=['Actual Result'], colnames=['Predicted Result'])
```

Out[45]:

Predicted Result	0	1
Actual Result		
0	11447	772
1	1792	1594

Evaluating the Logistic Regression Model

- Confusion matrix: Accuracy, Recall, Precision, F1 Score
- ROC (receiver operating characteristic) curve
  - The area under the ROC curve, or AUC (Area Under Curve)

<table>
<tr><td></td><td></td><td colspan="2">True condition</td><td></td><td colspan="2"></td></tr>
<tr><td></td><td>Total population</td><td>Condition positive</td><td>Condition negative</td><td>Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$</td><td colspan="2">Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$</td></tr>
<tr><td rowspan="2">Predicted condition</td><td>Predicted condition positive</td><td>True positive</td><td>False positive, Type I error</td><td>Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$</td><td colspan="2">False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$</td></tr>
<tr><td>Predicted condition negative</td><td>False negative, Type II error</td><td>True negative</td><td>False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$</td><td colspan="2">Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$</td></tr>
<tr><td></td><td></td><td>True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$</td><td>False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$</td><td>Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$</td><td rowspan="2">Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$</td><td rowspan="2">$F_1$ score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$</td></tr>
<tr><td></td><td></td><td>False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$</td><td>Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$</td><td>Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$</td></tr>
</table>

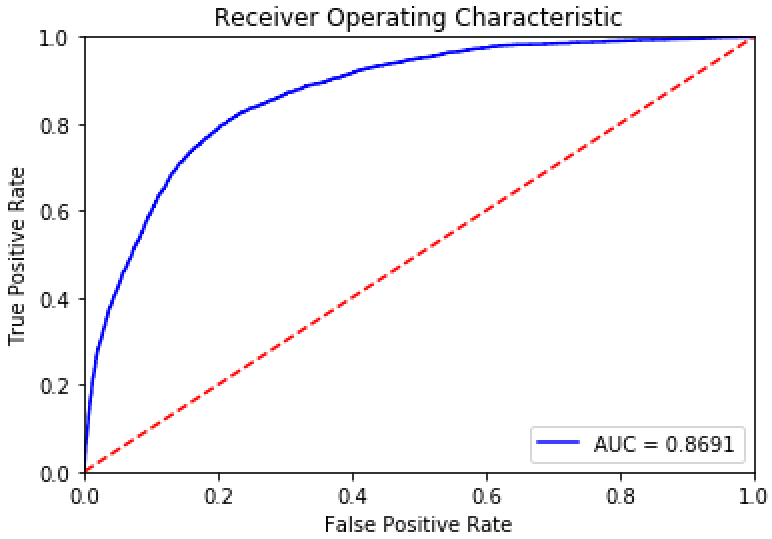
ROC

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

more information here: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

In [54]:
```
# calculate the fpr and tpr for all thresholds of the classification
probs = logreg.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)


plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.4f' % 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()
```



Confusion matrix, F1 score

In [68]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[11447   772]
 [ 1792  1594]]
```

In [70]:
```
from sklearn.metrics import f1_score
print(f1_score(y_test, y_pred, average='binary'))
```

```
0.5542420027816413
```

New in scikit-learn 0.22

- Conda
  - conda update -n base -c defaults conda
  - conda update --force conda
  - conda upgrade conda

```
In [1]: from sklearn.metrics import plot_confusion_matrix
```

```
In [23]: titles_options = [("Confusion matrix, without normalization", None),
                           ("Normalized confusion matrix", 'true')]
         classifier = logreg
         class_names = ["0", "1"]

         for title, normalize in titles_options:
             disp = plot_confusion_matrix(classifier, X_test, y_test,
                                          display_labels=class_names,
                                          cmap=plt.cm.Blues,
                                          normalize=normalize,
                                         values_format = ".2f")
             disp.ax_.set_title(title)

             print(title)
             print(disp.confusion_matrix)

         plt.show()
```

```
Confusion matrix, without normalization
[[11289   722]
 [ 1941  1653]]
Normalized confusion matrix
[[0.93988844 0.06011156]
 [0.54006678 0.45993322]]
```