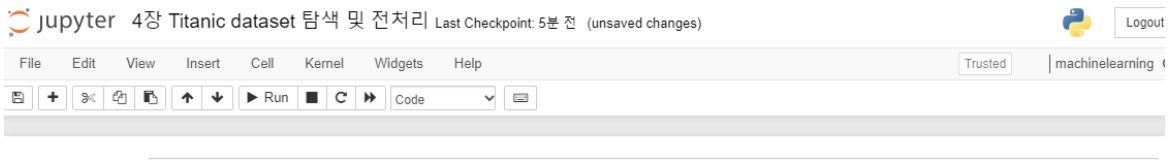
2주차 실습과제

2016314786 김호진

[4장 실습 – Titanic dataset 탐색 및 전처리]



In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.metrics import *
from pandas import DataFrame, Series

plt.style.use('seaborn')
sns.set(font_scale=2.5)
```

In [2]:
```
df_train = pd.read_csv('/Users/borus/Desktop/titanic/train.csv')
df_test = pd.read_csv('/Users/borus/Desktop/titanic/test.csv')
```

In [3]:
```
df_train.shape, df_test.shape
```

Out[3]:
```
((891, 12), (418, 11))
```

In [4]:
```
columns=df_train.columns
columns
```

Out[4]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [5]:
```
df_train.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [6]:
```
df_test.head()
```

Out[6]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [7]:
```
df_train.dtypes
```

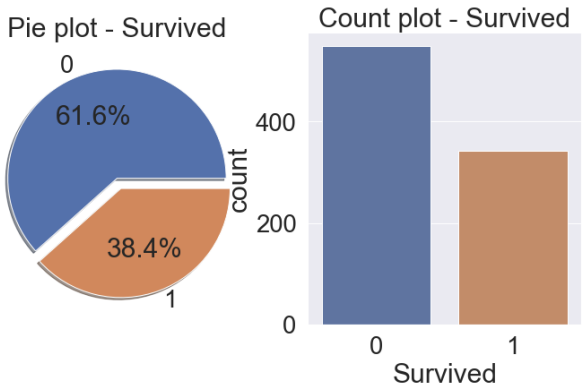
Out[7]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [8]:
```
df_train['Survived'].value_counts()
```

Out[8]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [9]:
```
f, ax = plt.subplots(1, 2, figsize =(12, 6))
df_train["Survived"].value_counts().plot.pie(explode=[0,0.1], autopct="%1.1f%%", ax=ax[0], shadow=True)
ax[0].set_title("Pie plot - Survived")
ax[0].set_ylabel('')
sns.countplot("Survived", data=df_train, ax=ax[1])
ax[1].set_title("Count plot - Survived")

plt.show()
```

```
C:\Users\borus\anaconda3\envs\machinelearning\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [10]:
```
df_train.isnull().sum()
```

Out[10]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [11]:
```
df_test.isnull().sum()
```

Out[11]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [12]:
```
train = df_train.drop(['Cabin', 'Embarked', 'Name', 'Ticket', 'PassengerId'],axis=1)
test = df_test.drop(['Cabin', 'Embarked', 'Name', 'Ticket'],axis=1)

train["Age"].fillna(train.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Age"].fillna(test.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Fare"].fillna(test.groupby("Sex")["Fare"].transform("median"), inplace=True)
```

In [13]:
```
train.isnull().sum()
```

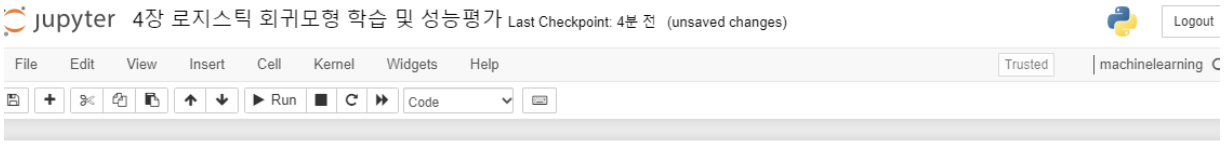
Out[13]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
dtype: int64
```

In [14]:
```
test.isnull().sum()
```

Out[14]:
```
PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
dtype: int64
```

In [15]:
```
sex_mapping = {"male": 0, "female": 1}
train['Sex'] = train['Sex'].map(sex_mapping)
test['Sex'] = test['Sex'].map(sex_mapping)

age_mean = train['Age'].mean()
age_std = train['Age'].std()
indexNames = train[train['Age'] < age_mean - 3*age_std].index
train.drop(indexNames , inplace=True)
indexNames = train[train['Age'] > age_mean + 3*age_std].index
train.drop(indexNames , inplace=True)
fare_mean = train['Fare'].mean()
fare_std = train['Fare'].std()
indexNames = train[train['Fare'] < fare_mean - 3*fare_std].index
train.drop(indexNames , inplace=True)
indexNames = train[train['Fare'] > fare_mean + 3*fare_std].index
train.drop(indexNames , inplace=True)
```

[4장 실습 – 로지스틱 회귀모형 학습 및 성능평가]



In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.metrics import *
from pandas import DataFrame, Series

plt.style.use('seaborn')
sns.set(font_scale=2.5)
df_train = pd.read_csv('/Users/borus/Desktop/titanic/train.csv')
df_test = pd.read_csv('/Users/borus/Desktop/titanic/test.csv')
train = df_train.drop(['Cabin', 'Embarked', 'Name', 'Ticket', 'PassengerId'],axis=1)
test = df_test.drop(['Cabin', 'Embarked', 'Name', 'Ticket'],axis=1)
train["Age"].fillna(train.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Age"].fillna(test.groupby("Sex")["Age"].transform("mean"), inplace=True)
test["Fare"].fillna(test.groupby("Sex")["Fare"].transform("median"), inplace=True)
sex_mapping = {"male": 0, "female": 1}
train['Sex'] = train['Sex'].map(sex_mapping)
test['Sex'] = test['Sex'].map(sex_mapping)
age_mean = train['Age'].mean()
age_std = train['Age'].std()
indexNames = train[train['Age'] < age_mean - 3*age_std].index
train.drop(indexNames , inplace=True)
indexNames = train[train['Age'] > age_mean + 3*age_std].index
train.drop(indexNames , inplace=True)
fare_mean = train['Fare'].mean()
fare_std = train['Fare'].std()
indexNames = train[train['Fare'] < fare_mean - 3*fare_std].index
train.drop(indexNames , inplace=True)
indexNames = train[train['Fare'] > fare_mean + 3*fare_std].index
train.drop(indexNames , inplace=True)
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split

X_train = train.drop('Survived', axis=1).values
target_label = train['Survived'].values
X_test = test.values
```

In [3]:
```
X_train.shape, X_test.shape
```

Out[3]:
```
((864, 6), (418, 7))
```

In [4]:
```
X_tr, X_vld, y_tr, y_vld = train_test_split(X_train, target_label, test_size=0.2, random_state=2020)
y_tr.shape, y_vld.shape
```

Out[4]:
```
((691,), (173,))
```

In [5]:
```
model = LogisticRegression()
model.fit(X_tr, y_tr)
prediction = model.predict(X_vld)
```

In [6]:
```
prediction
```

Out[6]:
```
array([1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [7]:
```
print('Number of people: {} \naccuracy: {:.2f}% '.format(y_vld.shape[0], 100 * accuracy_score(y_vld,prediction)))
```

```
Number of people: 173
accuracy: 78.03%
```

In [8]:
```
confusion_matrix(y_vld,prediction)
```

Out[8]:
```
array([[94, 24],
       [14, 41]], dtype=int64)
```

In [9]:
```
2f}% \nF1-score: {:.2f}% '.format(100*precision_score(y_vld,prediction),100*recall_score(y_vld,prediction),100*f1_score(y_vld,prediction)))
```

```
Precision: 63.08%
Recall: 74.55%
F1-score: 68.33%
```

In [10]:
```
list = []
for i in np.linspace(0,1,100):
    pred = model.predict_proba(X_vld)[:,1] > i
    cf_mtx = confusion_matrix(y_vld, pred)
    acc = accuracy_score(y_vld, pred)
    tpr = cf_mtx[0,0] / cf_mtx[0].sum()
    fpr = cf_mtx[1,0] / cf_mtx[1].sum()
    f1 = f1_score(y_vld, pred)
    list.append([i, acc, f1, tpr, fpr])

cut_off = DataFrame(list)
cut_off.columns = ["CUTOFF", "ACC", "F1", "TPR", "FPR"]
cut_off
```

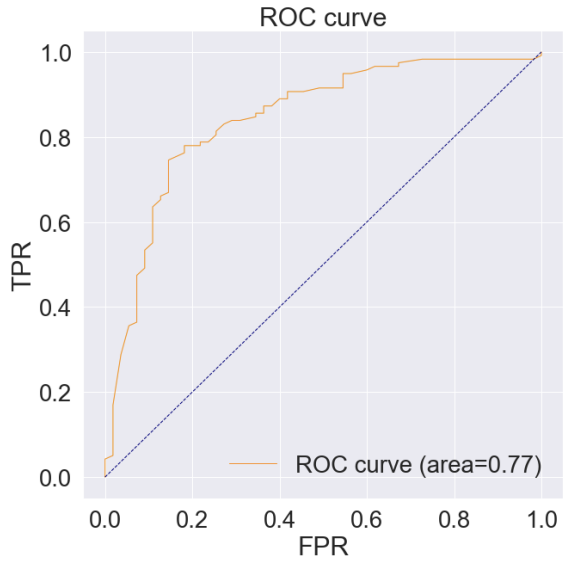
Out[10]:

	CUTOFF	ACC	F1	TPR	FPR
0	0.000000	0.317919	0.482456	0.000000	0.000000
1	0.010101	0.323699	0.484581	0.008475	0.000000
2	0.020202	0.323699	0.484581	0.008475	0.000000
3	0.030303	0.329480	0.486726	0.016949	0.000000
4	0.040404	0.329480	0.486726	0.016949	0.000000
...	...	...	...	...	...
95	0.959596	0.687861	0.100000	0.983051	0.945455
96	0.969697	0.687861	0.100000	0.983051	0.945455
97	0.979798	0.676301	0.034483	0.983051	0.981818
98	0.989899	0.676301	0.000000	0.991525	1.000000
99	1.000000	0.682081	0.000000	1.000000	1.000000

100 rows × 5 columns

In [11]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_vld, prediction)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(10,10))
plt.plot(cut_off["FPR"],cut_off["TPR"], color="darkorange", lw=1, label="ROC curve (area=%.2f)" %roc_auc)
plt.plot([0,1], [0,1], color='navy', lw=1, linestyle='--')
plt.title("ROC curve")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.legend(loc="lower right")
```

Out[11]:
```
<matplotlib.legend.Legend at 0x1e4e17d5250>
```



In [12]:
```
cut_off[cut_off["ACC"] == cut_off["ACC"].max()] #accuracy가 최대인 값
```

Out[12]:

	CUTOFF	ACC	F1	TPR	FPR
70	0.707071	0.803468	0.653061	0.90678	0.418182

In [13]:
```
cut_off_ACC_MAX = cut_off[cut_off["ACC"] == cut_off["ACC"].max()]["CUTOFF"][70]
cut_off_ACC_MAX
```

Out[13]:
```
0.7070707070707072
```

In [14]:
```
pred_ACC_MAX = model.predict_proba(X_vld)[:,1] > cut_off_ACC_MAX
```

In [15]:
```
confusion_matrix(y_vld,pred_ACC_MAX)
```

Out[15]:
```
array([[107,  11],
       [ 23,  32]], dtype=int64)
```

In [16]:
```
cut_off[cut_off["F1"] == cut_off["F1"].max()] #F1-score가 최대인 값
```

Out[16]:

	CUTOFF	ACC	F1	TPR	FPR
45	0.454545	0.791908	0.714286	0.779661	0.181818

In [17]:
```
cut_off_F1_MAX = cut_off[cut_off["F1"] == cut_off["F1"].max()]["CUTOFF"][45]
cut_off_F1_MAX
```

Out[17]:
```
0.4545454545454546
```

In [18]:
```
pred_F1_MAX = model.predict_proba(X_vld)[:,1] > cut_off_F1_MAX
```

In [19]:
```
confusion_matrix(y_vld,pred_F1_MAX)
```

Out[19]:
```
array([[92, 26],
       [10, 45]], dtype=int64)
```

[5장 실습 – 나이브베이즈]

jupyter 5장 나이브베이즈 Last Checkpoint: 7분 전 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted machinelearning

Run Code

In [1]:
```
import warnings
warnings.filterwarnings(action='ignore')
```

In [2]:
```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [3]:
```
iris = load_iris()
iris_frame = pd.DataFrame(data=np.c_[iris['data'], iris['target']],columns = iris['feature_names'] + ['target'])
iris_frame['target'] = iris_frame['target'].map({1:"versicolor",0:"setosa",2:"virginica"})
X = iris_frame.iloc[:,:-1]
Y = iris_frame.iloc[:,[-1]]
iris_frame
```

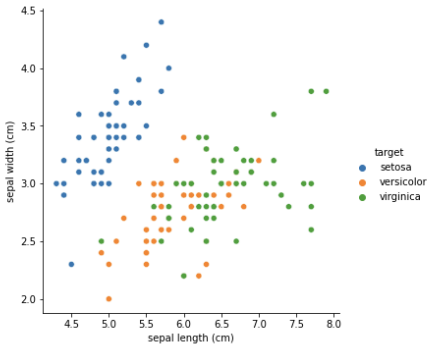
Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

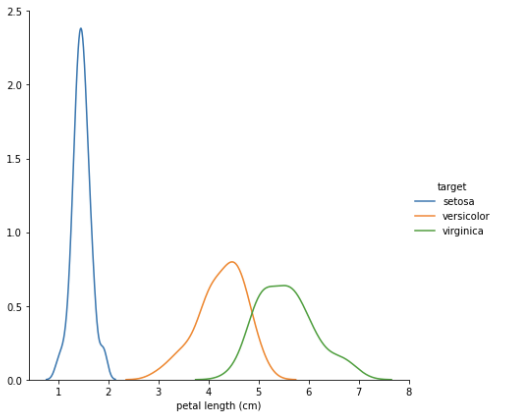
In [4]:
```
import seaborn as sn

sn.pairplot(iris_frame, x_vars=["sepal length (cm)"], y_vars=["sepal width (cm)"], hue = "target", size=5)
```

Out[4]: <seaborn.axisgrid.PairGrid at 0x13b35ec3ca0>



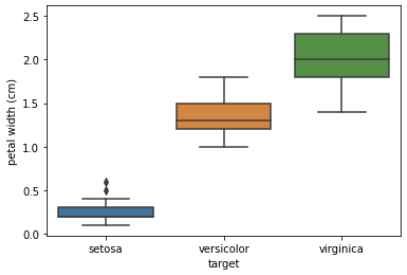
In [5]:
```
sn.FacetGrid(iris_frame, hue="target", size=6).map(sn.kdeplot, "petal length (cm)").add_legend()
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x13b3795cee0>



In [6]:
```
sn.boxplot(x="target", y="petal width (cm)", data=iris_frame)
```

Out[6]:
```
<AxesSubplot:xlabel='target', ylabel='petal width (cm)'>
```



In [7]:
```
import matplotlib.colors as colors
from sklearn.naive_bayes import GaussianNB
df1 = iris_frame[["sepal length (cm)" , "sepal width (cm)", "target" ]]
X = df1.iloc[:,0:2]
Y = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()
NB=GaussianNB()
NB.fit(X,Y)
N=100
```

In [8]:
```
X_= np.linspace(4,8,N)
Y_= np.linspace(1.5,5,N)
X_,Y_= np.meshgrid(X_,Y_)

color_list= ['Blues','Greens','Reds']
my_norm = colors.Normalize(vmin=-1,vmax=1)
g= sn.FacetGrid(iris_frame, hue = "target", size =10,
                palette = 'colorblind').map(plt.scatter, "sepal length (cm)","sepal width (cm)",).add_legend()

my_ax = g.ax
zz = np.array( [NB.predict ([[xx,yy]]) [0] for xx,yy in zip(np.ravel(X_),np.ravel(Y_))])
Z=zz.reshape(X_.shape)

my_ax.contourf(X_,Y_,Z,2,alpha=.1, colors = ('blue','green','red'))
my_ax.contour(X_,Y_,Z,2,alpha=1, colors = ('blue','green','red'))

my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('Gaussian Naive Bayes boundaries')
```

Out[8]:
```
Text(0.5, 1.0, 'Gaussian Naive Bayes boundaries')
```