Machine Learning with Python

Life is too short, You need Python

실습 내용

- Titanic 데이터로 모델링합니다.
- Decision Tree 알고리즘으로 모델링합니다.

1.환경 준비

- 기본 라이브러리와 대상 데이터를 가져와 이후 과정을 준비합니다.

In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format='retina'
```

In [2]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/jangrae/csv/master/titanic_train.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 충분히 이해할 수 있도록 다양한 탐색 과정을 수행합니다.

In [3]:
```
# 상위 몇 개 행 확인
data.head()
```

Out[3]:

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

In [4]:
```
# 기술통계 확인
data.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:
```
# Survived 확인
data['Survived'].value_counts()
```

Out[5]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

In [6]:
```
# NaN 값 확인
data.isnull().sum()
```

Out[6]:
```
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 [7]:
```
# 상관관계 확인
data.corr(numeric_only=True)
```

Out[7]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

3.데이터 준비

- 전처리 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.

1) 변수 제거

- 분석에 의미가 없다고 판단되는 변수는 제거합니다.

In [8]:
```
# 제거 대상: PassengerId, Name, Ticket, Cabin
drop_cols = ['PassengerId', 'Name', 'Ticket', 'Cabin']

# 변수 제거
data.drop(drop_cols, axis=1, inplace=True)

# 확인
data.head()
```

Out[8]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

2) 결측치 처리

- 결측치가 있으면 제거하거나 적절한 값으로 채웁니다.

In [9]:
```
# Age 결측치를 중앙값으로 채우기
age_median = data['Age'].median()
data['Age'].fillna(age_median, inplace=True)
```

In [10]:
```
# Embarked 최빈값으로 채우기
emb_freq = data['Embarked'].mode()[0]
data['Embarked'].fillna(emb_freq, inplace=True)
```

3) x, y 분리

- 우선 target 변수를 명확히 지정합니다.
- target을 제외한 나머지 변수들 데이터는 x로 선언합니다.
- target 변수 데이터는 y로 선언합니다.
- 이 결과로 만들어진 x는 데이터프레임, y는 시리즈가 됩니다.
- 이후 모든 작업은 x, y를 대상으로 진행합니다.

In [11]:
```
# target 확인
target = 'Survived'

# 데이터 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

4) 가변수화

- 범주형 변수에 대한 가변수화를 진행합니다.

In [12]:
```
# 가변수화 대상: Pclass, Sex, Embarked
dumm_cols = ['Pclass', 'Sex', 'Embarked']

# 가변수화
x = pd.get_dummies(x, columns=dumm_cols, drop_first=True, dtype=int)

# 확인
x.head()
```

Out[12]:

	Age	SibSp	Parch	Fare	Pclass_2	Pclass_3	Sex_male	Embarked_Q	Embarked_S
0	22.0	1	0	7.2500	0	1	1	0	1
1	38.0	1	0	71.2833	0	0	0	0	0
2	26.0	0	0	7.9250	0	1	0	0	1
3	35.0	1	0	53.1000	0	0	0	0	1
4	35.0	0	0	8.0500	0	1	1	0	1

5) 학습용, 평가용 데이터 분리

- 학습용, 평가용 데이터를 적절한 비율로 분리합니다.
- 반복 실행 시 동일한 결과를 얻기 위해 random_state 옵션을 지정합니다.

In [13]:
```
# 모듈 불러오기
from sklearn.model_selection import train_test_split

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

4.모델링

- 본격적으로 모델을 **선언**하고 **학습**하고 **평가**하는 과정을 진행합니다.

In [14]:
```
# 1단계: 불러오기
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

In [43]:
```
# 2단계: 선언하기
model = DecisionTreeClassifier(max_depth=5 ,random_state=1) #독립 변수 제한 # 같은 걸 고를 수
```

In [44]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[44]:
```
DecisionTreeClassifier(max_depth=5, random_state=1)
```

In [45]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

In [50]:
```
# 5단계 평가하기
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[137  16]
 [ 46  69]]
              precision    recall  f1-score   support

           0       0.75      0.90      0.82       153
           1       0.81      0.60      0.69       115

    accuracy                           0.77       268
   macro avg       0.78      0.75      0.75       268
weighted avg       0.78      0.77      0.76       268
```

5.기타

- 기타 필요한 내용이 있으면 진행합니다.

1) 트리 시각화

- Decision Tree는 시각화를 통해 모델이 어떻게 작동하는 지 확인할 수 있습니다.
- 여러 가지 시각화 방법이 있지만 Graphviz 패키지를 사용해봅니다.
- 사전에 Graphviz 패키지 설치 및 운영체제 환경 설정이 진행되어야 합니다.

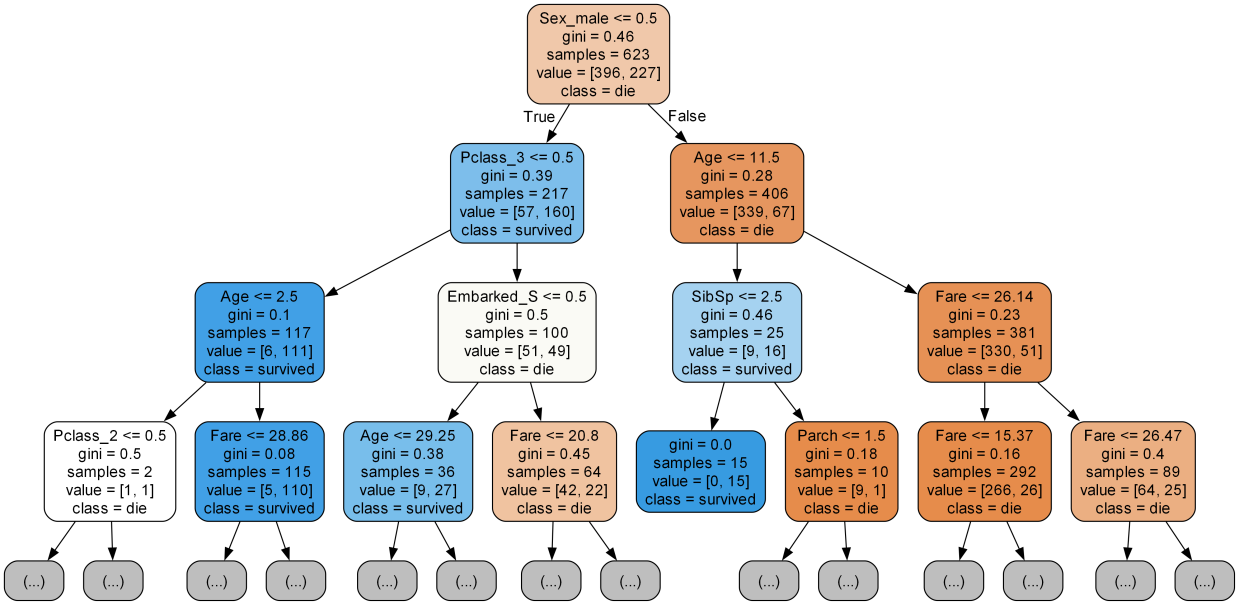
In [47]:
```
# 시각화 모듈 불러오기
from sklearn.tree import export_graphviz
from IPython.display import Image

# 이미지 파일 만들기
export_graphviz(model,                                  # 모델 이름
                out_file='tree.dot',                    # 파일 이름
                feature_names=list(x),                  # Feature 이름
                class_names=['die', 'survived'],        # Target Class 이름 (분류인 경우만 지정
                rounded=True,                           # 둥근 테두리
                precision=2,                            # 불순도 소숫점 자리수
                max_depth=3,                            # 실제로 표시할 트리 깊이
                filled=True)                            # 박스 내부 채우기

# 파일 변환
!dot tree.dot -Tpng -otree.png -Gdpi=300

# 이미지 파일 표시
Image(filename='tree.png')
```

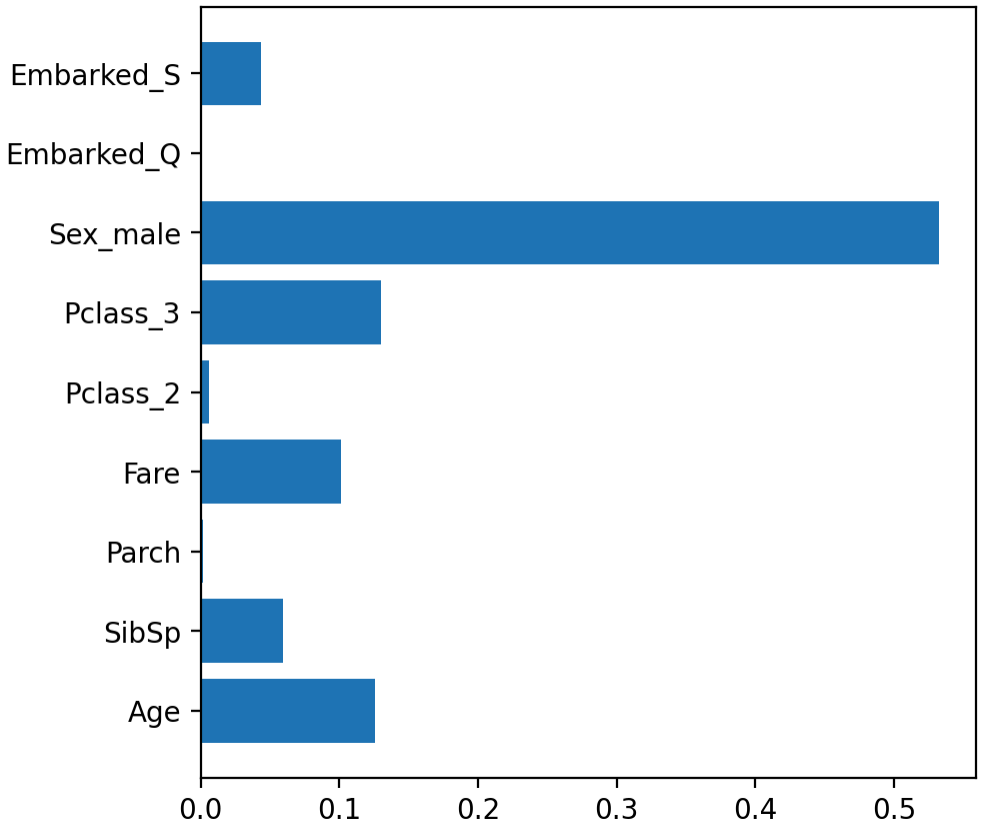
Out[47]:



2) 변수 중요도 시각화

- 변수 중요도를 시각화해 봅니다.
- feature*importances* 속성이 변수 중요도입니다.
- 세로 막대 보다는 가로 막대 그래프로 보면 좋습니다.

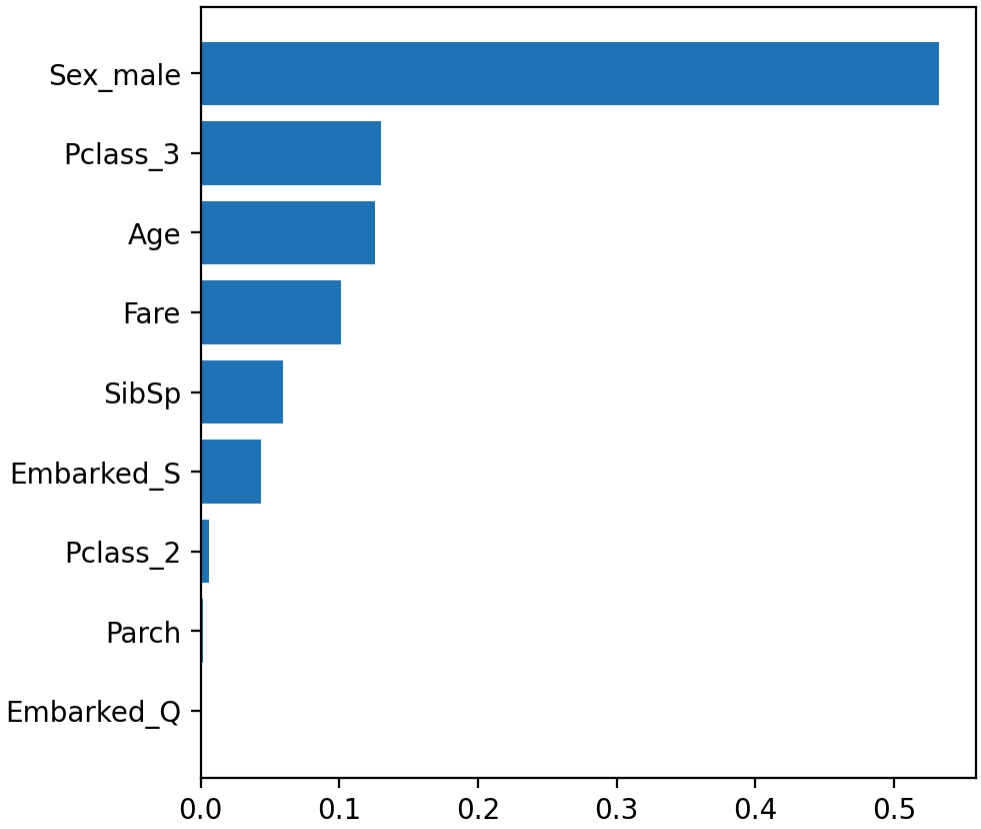
In [48]:
```
# 변수 중요도
plt.figure(figsize=(5, 5))
plt.barh(y=list(x), width=model.feature_importances_) # feature_importances_ 속성이 변수 중요도
plt.show()
```



- 필요하면 다음과 같이 중요도를 기준으로 정렬해 시각화합니다.

In [49]:
```
# 데이터프레임 만들기
df = pd.DataFrame()
df['feature'] = list(x)
df['importance'] = model.feature_importances_
df.sort_values(by='importance', ascending=True, inplace=True)

# 시각화
plt.figure(figsize=(5, 5))
plt.barh(df['feature'], df['importance'])
plt.show()
```



In [ ]: