Kaggleコンペ　～タイタニック号沈没事故の生存予測～

はじめに

機械学習の練習として、タイタニックの生存予測というものがあります。タイタニック号の乗客のデータセットをもとに機械学習モデルを作成し乗客の生死を判定するものです。
有名なコミュニティにkaggleというものがあります。そこでは様々なデータが公開され、お題のデータセットに対してコンペと呼ばれる予測モデル作成を競い合う大会のようなものが行われていたりと、機械学習に関する有用なコミュニティです。タイタニック号の生存予測もkaggleのコンペに用いられ、練習問題として広く知られています。今回はこれを行っていきます。

分析するデータ

kagglのタイタニック号の生存予測のページにあるデータセットを使用

実行環境

パソコン：macbook pro
開発環境：Google Coraboratory
言語：Python
ライブラリ：pandas, Numpy, Matplotlib

分析の流れ

1.データの読み込み
2.データの内容確認
3.前処理
4.モデルの選択
5.学習
6.学習結果と分析の評価

実行したコードとその結果

ライブラリをインポート

```
#ライブラリをインポート
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
import pandas as pd
```

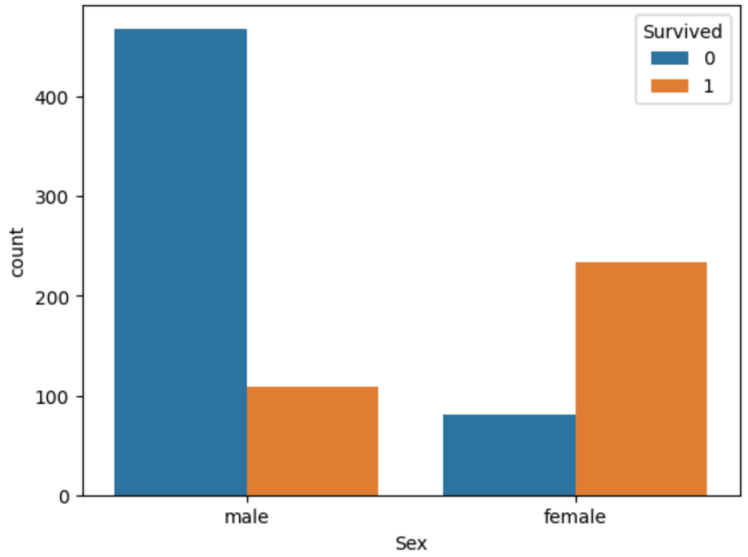
データの読み込み

```
train = pd.read_csv('/kaggle/input/titanic/train.csv')
test = pd.read_csv('/kaggle/input/titanic/test.csv')
```

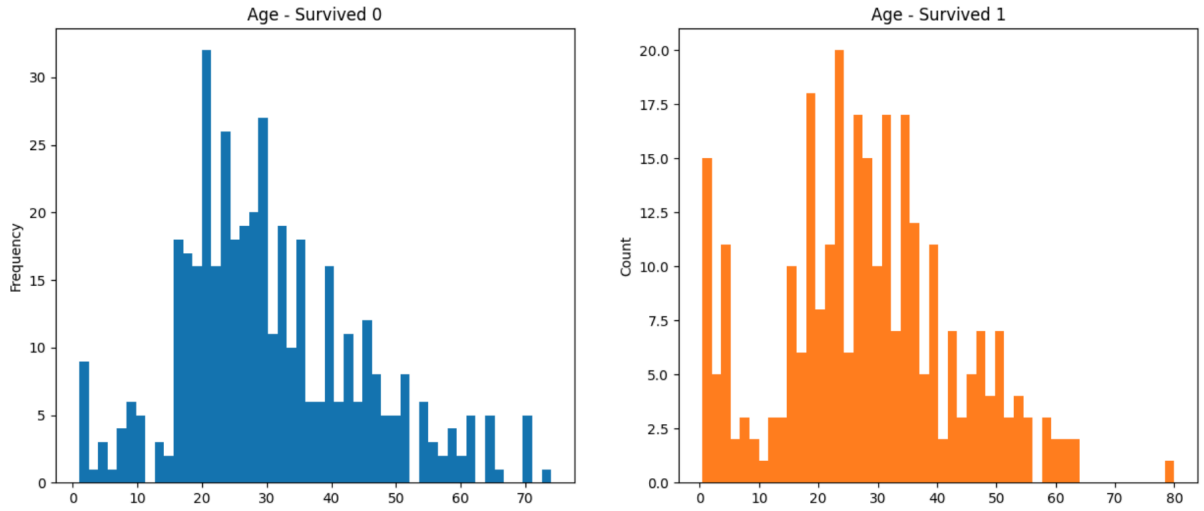
データの確認
データの可視化と確認

```
#Sex(性別)
sns.countplot(x='Sex', hue='Survived', data=train)
```

```
<Axes: xlabel='Sex', ylabel='count'>
```



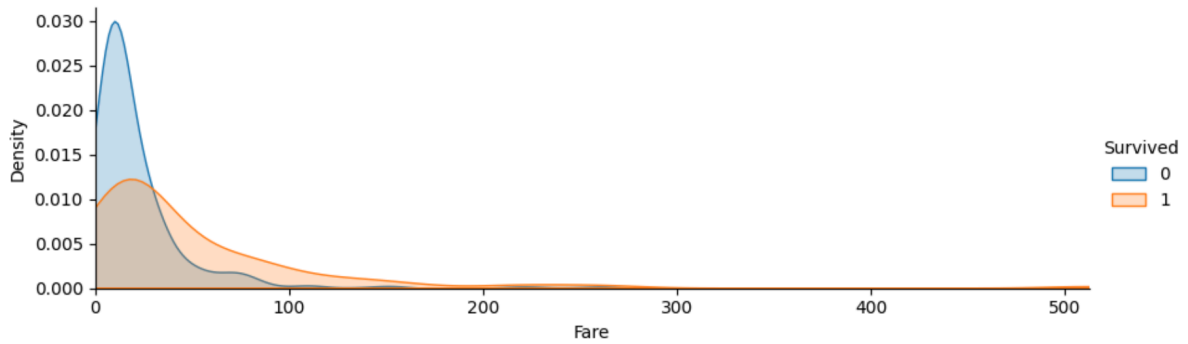

```
#Age(年齢)
cmap = plt.get_cmap('tab10')
fig, ax = plt.subplots(1, 2, figsize=(15, 6))
train[train['Survived']==0]['Age'].plot(kind='hist', bins=50, title='{} - {} 0'.format('Age', 'Survived'),
train[train['Survived']==1]['Age'].plot(kind='hist', bins=50, title='{} - {} 1'.format('Age', 'Survived'),
plt.ylabel('Count')
plt.show()
```



```
#Pclassごとの平均年齢の確認
train.groupby('Pclass').describe()['Age']
```

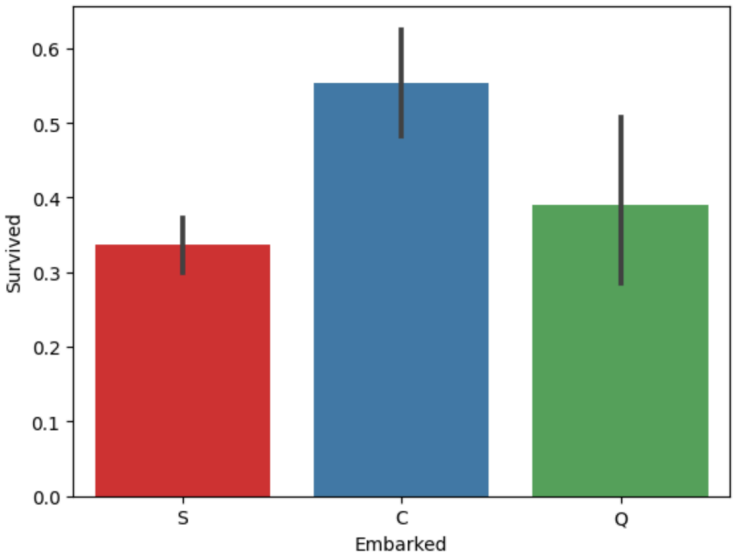
Pclass	count	mean	std	min	25%	50%	75%	max
1	186.0	38.233441	14.802856	0.92	27.0	37.0	49.0	80.0
2	173.0	29.877630	14.001077	0.67	23.0	29.0	36.0	70.0
3	355.0	25.140620	12.495398	0.42	18.0	24.0	32.0	74.0

```
#Fare(料金)
fare_s = sns.FacetGrid(train, hue='Survived',aspect=3)
fare_s.map(sns.kdeplot,'Fare',shade= True)
fare_s.set(xlim=(0, train['Fare'].max()))
fare_s.add_legend()
```




```
#Embarked(出港地)
sns.barplot(x='Embarked', y='Survived', data=train, palette='Set1')
```

```
<Axes: xlabel='Embarked', ylabel='Survived'>
```



分析したデータから得た得た情報

- 「Age(年齢)」に欠損値がある。
- チケットのランクが高い程、生存率が高い。
- 「Name(姓名)」中に記載されている敬称(title)を利用して、年齢に関するデータの欠損値を埋めることが出来る可能性がある。
- 女性の方が、生存率が高い
- 「Pclass(チケットのランク)」ごとに、平均年齢が異なる。
- 「SibSp(同乗している兄弟、配偶者の数)」が大きいと、生存率が低下する傾向がありそう。
- 「Fare(料金)」が低いほど、生存率も低いよう。
- 「Embarked(出港地)」により、生存率に差がある模様。

データの前処理

文字列型を数値化
年齢をPclass毎に、中央値で穴埋め
「Fare」の欠損値の1つを中央値、「Embarked」の欠損値の2つを最頻値で穴埋め

```
#前処理
train=train.replace({'male': 0 ,'female': 1 })
test=test.replace({'male': 0 ,'female': 1 })

train.loc[train['Age'].isnull(), 'Age'] = train.groupby('Pclass')['Age'].transform('median')
test.loc[test['Age'].isnull(), 'Age'] = test.groupby('Pclass')['Age'].transform('median')

train['Embarked'] = train['Embarked'].fillna(train['Embarked'].mode())

test['Fare'] = test['Fare'].fillna(test['Fare'].median())
```

敬称の再分類化

```
#称から性別と年齢が推定
train['Title_tr'] = train['Name'].map(lambda x: x.split(', ')[1].split('. ')[0])
train['Title_tr'] = train['Title_tr'].replace(['Dr','Rev','Mlle','Major','Col','the Countess',
                                               'Capt','Ms','Sir','Lady','Mme','Don','Jonkheer'], "Others")
train['Title_tr'] = train['Title_tr'].map( {'Master': 0 , 'Miss':1 , 'Mr':2, 'Mrs':3, 'Others':4})

test['Title_te'] = test['Name'].map(lambda x: x.split(', ')[1].split('. ')[0])
test['Title_te'] = test['Title_te'].replace(['Col','Rev','Ms','Dr','Dona'], 'Others')
test['Title_te'] = test['Title_te'].map( {'Master': 0 , 'Miss':1 , 'Mr':2, 'Mrs':3, 'Others':4})
```

ダミー変数化

```
#ダミー変数化
train_embarked_dum = pd.get_dummies(train['Embarked'], dtype='uint8')
train_add_dum = pd.concat([train,train_embarked_dum], axis=1 )
train_add_dum.head()

test_embarked_dum = pd.get_dummies(test['Embarked'], dtype='uint8')
test_add_dum = pd.concat([test,test_embarked_dum], axis=1 )
test_add_dum.head()
```

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

予測モデルの構築

```
#予測モデル作成
from sklearn.ensemble import RandomForestClassifier
#学習データの特徴量と目的変数を取得
train_features = train_add_dum[['Pclass', 'Sex', 'Age', 'Fare', 'Title_tr', 'C', 'Q', 'S']].values
train_target = train['Survived'].values
```

ハイパーパラメータのチューニング

```
#ハイパーパラメータをチューニング
train_features_tr, train_features_va, train_target_tr, train_target_va = train_test_split(train_features,
                                    train_target, test_size = 0.2, random_state=0)

from sklearn.model_selection import GridSearchCV
rf_model = RandomForestClassifier(random_state=0)

max_depth = [4, 5, 6, 7, 8, 9, 10]
n_estimators = [120, 130, 140, 150, 160]
max_features = [6, 7, 8]
params = {'max_depth':max_depth, 'n_estimators':n_estimators, 'max_features':max_features}

grid_cv = GridSearchCV(estimator=rf_model, param_grid=params, cv=5)
grid_cv.fit(train_features_tr, train_target_tr)

display(grid_cv.best_params_, grid_cv.best_score_,grid_cv.score(train_features_va, train_target_va))
```

```
{'max_depth': 8, 'max_features': 7, 'n_estimators': 130}
0.8384615384615385
0.8491620111731844
```

モデルを作成の上、学習

```
model = RandomForestClassifier(n_estimators = 130, max_depth=8, max_features=7, random_state=0)
model.fit(train_features, train_target)
```

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=8, max_features=7, n_estimators=130,
                       random_state=0)
```

作成したモデルで目的変数を予測

```
test_features = test_add_dum[['Pclass', 'Sex', 'Age', 'Fare','Title_te', 'C', 'Q', 'S']].values
predict_test_target = model.predict(test_features)
```

予測モデルのスコア