报告

学号: 3019213043　　姓名:刘京宗　　班级:软工 5 班

1. 目标

使用 python 完成对灾难数据集 Titanic Machine Learning 的全过程数据分析和预测。

2. 任务

按照以下实验步骤完成实验报告。

Data Dictionary

Variable	Definition	type	Explain
PassengerId	Id of passengers	int	Train data: 1-891 Test data: 892-1309
Survived	Survived or not	bool	0:No 1:Yes
Pclass	Ticket class	int	1: 1st 2: 2nd 3:3rd
Name	Name of passengers	string	
Sex	Sex of passengers	string	Male female
Age	Age in years	int	
SibSp	# of siblings / spouses aboard the Titanic	int	
ParCh	# of parents / children aboard the Titanic	int	
Ticket	Ticket number	string	

Fare	Passenger fare	string	
Cabin	Cabin number	string	
Embarked	embarked Port of Embarkation	string	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

1. pclass: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

2. age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

3. sibsp: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

4. Spouse = husband, wife (mistresses and fiancés were ignored)

5. parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

2.1 数据清洗

并不是每个变量都是有用的，也不是每个变量都适合建立模型，所以我们需要从原有的变量中提取信息来创建新的特征。

1) Name 变量

在 Name 变量中，每个人都有一个 Passenger 头衔。根据这个标题，我们可以用它来代替 Name 变量。比如：我们可以将标题进行划分，将低频标题合并成一个类。

For example: 'Dona', 'Lady', 'the Countess','Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer' can be merged.

Mlle <- 'Miss'

Ms <- 'Miss'

'Mme'<-'Mrs'

```
# Load in the train and test datasets
train = pd.read_csv('实验 5/train.csv')
test = pd.read_csv('实验 5/test.csv')
# Store our passenger ID for easy access
PassengerId = test['PassengerId']
print(train.head(3))

full_data = [train, test]


# Define function to extract titles from passenger names
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    # If the title exists, extract and return it.
    if title_search:
        return title_search.group(1)
    return ""


# Create a new feature Title, containing the titles of passenger names
```

```
for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)
# Group all non-common titles into one single grouping "Rare"
for dataset in full_data:
    dataset['Title'] = dataset['Title'].replace(
        ['Lady', 'Countess', 'Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
print(train.head(3))
```

运行结果如下：

	PassengerId	Survived	Pclass	...	Cabin	Embarked	Title
0	1	0	3	...	NaN	S	Mr
1	2	1	1	...	C85	C	Mrs
2	3	1	3	...	NaN	S	Miss

2) Family size 变量:

我们可以在 SibSp 和 Parch 变量的基础上创建一个新的 Family size 变量。用柱状图列出家庭数量与存活率之间的关系。并对变量进行分化，例如大小{1,2-4,>4}或者其他.

```
for dataset in full_data:
    dataset['Family_num'] = 1 + dataset['SibSp'] + dataset['Parch']
    dataset['FamilySize'] = ''
for dataset in full_data:
    for i in range(len(dataset)):
        if dataset['Family_num'][i] == 1:
            dataset['FamilySize'][i] = '1'
        elif 1 < dataset['Family_num'][i] <= 4:
            dataset['FamilySize'][i] = '2-4'
        else:
            dataset['FamilySize'][i] = '>4'
print(train.head(3))
```

运行结果如下：

	PassengerId	Survived	Pclass	...	Title	Family_num	FamilySize
0	1	0	3	...	Mr	2	2-4
1	2	1	1	...	Mrs	2	2-4
2	3	1	3	...	Miss	1	1

3) More variables:

在 Cabin 变量中，我们可以得到甲板信息，也就是 Cabin 的第一个字母。

```
for dataset in full_data:
    for i in range(len(dataset)):
        train['Cabin'][i] = str(train['Cabin'][i])
for dataset in full_data:
    for i in range(len(dataset)):
        if dataset['Cabin'][i] == 'nan':
            dataset['Cabin'][i] = 'none'
        else:
            dataset['Cabin'][i] = str(dataset['Cabin'][i])[0]
print(train['Cabin'].head(5))
```

运行结果如下：

```
0    none
1       C
2    none
3       C
4    none
Name: Cabin, dtype: object
```

2.2 缺省值

方法:

1) 删除整行或整列

2）使用平均值

3）使用中位数

4) 预测

例如如下，你可以使用合适的方法来填充缺失的值：

a. missing Embarked

   use Median? Or same Passenger class and fare have same Embarked?

b. Missing Fare

c. Missing Age

Prediction method

……

a. 在 Embarked 特征中丢失了两个值，将样本中出现最多的值填进去。

```
freq_port = train.Embarked.dropna().mode()[0]
print(freq_port)
for dataset in full_data:
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)
```

b. 使用中位数进行补全 Fare

```
test['Fare'].fillna(test['Fare'].dropna().median(), inplace=True)
```

c. 首先在 Sex 中，用‘0’代替 male，‘1’代替 female。然后根据 Sex(0 or 1)和 Pclass(1, 2, 3)两个特征来猜测 Age。

```
for dataset in full_data:
    dataset['Sex'] = dataset['Sex'].map({'female': 1, 'male': 0}).astype(int)
guess_ages = np.zeros((2, 3))
for dataset in full_data:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & (dataset['Pclass'] == j + 1)]['Age'].dropna()
            age_guess = guess_df.median()
            guess_ages[i, j] = int(age_guess / 0.5 + 0.5) * 0.5
    for i in range(0, 2):
        for j in range(0, 3):
            dataset.loc[(dataset.Age.isnull()) & (dataset.Sex == i) & (dataset.Pclass == j + 1),
'Age'] = guess_ages[
                    i, j]

    dataset['Age'] = dataset['Age'].astype(int)

train['AgeBand'] = pd.cut(train['Age'], 5)
train[['AgeBand', 'Survived']].groupby(['AgeBand'],
as_index=False).mean().sort_values(by='AgeBand', ascending=True)

for dataset in full_data:
    dataset.loc[dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[dataset['Age'] > 64, 'Age']

train = train.drop(['AgeBand'], axis=1)
```

2.3 特征工程

创建‘children’ and ‘mother’ 变量

Variable	Definition
children	0: child :Age<18 1: adult: Age>18
Mother	0: mother: female, adult, have one or more children, title is not ‘Miss’ 1: not mother

```
for dataset in full_data:
    dataset['Children'] = 0
    for i in range(len(dataset)):
        if dataset['Age'][i] < 18:
            dataset['Children'][i] = 0
        else:
            dataset['Children'][i] = 1

for dataset in full_data:
    dataset['Mother'] = 1
    for i in range(len(dataset)):
        if dataset['Age'][i] >= 18 and dataset['Title'][i] != 'Miss' and dataset['Sex'][i] == 'female'
and \
                dataset['Parch'][i] > 0:
            dataset['Mother'][i] = 0
```

最后用数字代替部分标签并且删除无用列：

```
for dataset in full_data:
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)
    dataset['Embarked'] = dataset['Embarked'].map({'S': 0, 'C': 1, 'Q': 2}).astype(int)
train_df = train.drop(['Ticket', 'Cabin', 'PassengerId', 'Parch', 'SibSp', 'Name'], axis=1)
test_df = test.drop(['Ticket', 'Cabin', 'PassengerId', 'Parch', 'SibSp', 'Name'], axis=1)
```

最终训练集数据为下表（前 6 列）：

Survived	Pclass	Sex	Age	Fare	Embarked	Title	Family_num	Children	Mother
0	3	0	1	7.25	0	1	2	1	1
1	1	1	2	71.2833	1	3	2	1	1
1	3	1	1	7.925	0	2	1	1	1
1	1	1	2	53.1	0	3	2	1	1
0	3	0	2	8.05	0	1	1	1	1
0	3	0	1	8.4583	2	1	1	1	1

2.4 模型构建和预测

1.使用训练数据建立模型

2.利用测试数据验证模型。

3.使用错误率评价指标。

使用多种不同方法对结果进行预测。

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, Perceptron, SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier

train = pd.read_csv('实验 5/train2.csv')
test = pd.read_csv('实验 5/test2.csv')
X_train = train.drop("Survived", axis=1)
Y_train = train["Survived"]

X_test = test.drop("PassengerId", axis=1)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/knn_predict.csv', index=False)
```

```
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/svc_predict.csv', index=False)

gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/gaussian_predict.csv', index=False)

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/decision_tree_predict.csv', index=False)

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/random_forest_predict.csv', index=False)

perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
```

```
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/perceptron_predict.csv', index=False)

sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": Y_pred
})
submission.to_csv('实验 5/sgd_predict.csv', index=False)
```

最终的结果如下，随机森林分类的结果最好。其中 random_forest_predict-2.csv 相较于 random_forest_predict.csv 的区别是 random_forest_predict-2.csv 的训练和测试的过程中没有使用‘children’ and ‘mother’ 变量。这表明 2.3 中的特征工程两个变量的添加对最后预测结果是有部分积极效果的。

Submission and Description	Public Score
random_forest_predict.csv 18 minutes ago by Liujingzong add submission details	0.77033
random_forest_predict-2.csv 2 minutes ago by Liujingzong add submission details	0.76315
decision_tree_predict.csv 18 minutes ago by Liujingzong add submission details	0.76315
gaussian_predict.csv 16 minutes ago by Liujingzong add submission details	0.74880
knn_predict.csv 13 minutes ago by Liujingzong add submission details	0.69138
svc_predict.csv 12 minutes ago by Liujingzong add submission details	0.66746
perceptron_predict.csv 14 minutes ago by Liujingzong add submission details	0.54784
sgd_predict.csv 13 minutes ago by Liujingzong add submission details	0.54306