notebooke9b25a365d

June 16, 2023

[ ]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

0.1 read train_data

As explained in the lecture for week 8, if emphasis is on prediction, data is divided into two (or three) parts. A model learned from the training data is used to check the prediction accuracy with the set-aside test data.

On Kaggle, training and test data are provided separately. Models are typically trained using the former and used to make predictions on the test data. The results are then often submitted as competition deliverables (submissions).

Kaggle

[ ]:
```
train = pd.read_csv('/kaggle/input/titanic/train.csv')
train.head()
```

```
[ ]: train.columns
```

0.2 summarize features

Check various statistical measures.

```
[ ]: train.describe()
```

0.3 check the ratio of Survived

First, confirm the survival rate of the training data.

```
[ ]: train['Survived'].value_counts()
```

0.4 Confirm the survival rates by gender

```
[ ]: train.groupby('Sex')['Survived'].mean()
```

```
[ ]: train.pivot_table('Survived', index='Sex', columns='Pclass')
```

0.5 Divide the age into groups at 18 years old and observe the survival rate

```
[ ]: Age = pd.cut(train['Age'], [0, 18, 80])
     train.pivot_table('Survived', ['Sex', Age], 'Pclass')
```

0.6 Check for missing values

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

0.7 plot some features

```
[ ]: import matplotlib.pyplot as plt
     import seaborn as sns
     sns.countplot(x = 'Sex', hue = "Survived", data = train)
     plt.legend(loc = "upper right", title = "Survived ~ Sex")
```

```
[ ]: sns.stripplot(x='Sex', y='Age', data=train, hue='Survived')
```

0.8 if family members are on board, it can be observed that it likely affects the survival rate.

Brothers and sisters.

```
[ ]: sns.countplot(x = 'SibSp', hue = "Survived", data = train)
     plt.legend(loc = "upper right", title = "Survived ~ Sibsp")
```

Paents or children.

```
[ ]: sns.countplot(x = 'Parch', hue = "Survived", data = train)
     plt.legend(loc = "upper right", title = "Survived ~ Parch")
```

Add a new variable (feature) that indicates whether there was one or more sibling, parent, or child.

```
[ ]: train['Alone']=np.where((train["SibSp"]+train["Parch"])>0,0,1)
     train.drop(['SibSp', 'Parch'], axis=1, inplace=True)
```

0.9 The port of embarkation does not seem to be related to the survival rate.

```
[ ]: sns.countplot(x = 'Embarked', hue = "Survived", data = train)
     plt.legend(loc = "upper right", title = "Survived ~ Embarked")
```

```
[ ]: train.drop('Embarked', axis=1, inplace=True)
```

0.10 The ticket fare is related to the cabin class, but it seems unusable due to the lack of uniformity in the input units

```
[ ]: train['Ticket']
```

```
[ ]: train.drop('Ticket', axis=1, inplace=True)
```

```
[ ]: train.columns
```

0.11 The passenger names, numbers, and cabin numbers are considered unnecessary. (However, there are data scientists who have utilized the presence of shared surnames as a feature.)

[ ]:
```
train.drop(['PassengerId','Name','Cabin'], axis=1, inplace=True)
```

0.12 Since it is believed that the fare is related to the cabin class, it is possible to consider excluding it to avoid multicollinearity. However, when checking the correlation, it seems to be small, so it will be kept as it is.

[ ]:
```
train.corr()
```

[ ]:
```
train.columns
```

0.13 Preprocessing:

- Focus on features that are likely to have an impact on survival.
- Transform the data to make it suitable for applying machine learning techniques.

Although there are many missing values for age, since it is an important variable, we will not remove the data but instead impute the missing values with the median.

However, since we will later impute missing values in the test data using the statistics from the training data, let's check the median of the age in the test data beforehand.

[ ]:
```
trainMedian = train["Age"].median(skipna=True)
train["Age"].fillna(trainMedian, inplace=True)
```

0.14 The cabin class is represented with numerical values of 1, 2, and 3. However, if left as is, it may be analyzed as if 3 is better (higher) than 1. To avoid this, it should be converted into a categorical variable."

The same can be said for Sex as well.

[ ]:
```
train['Pclass'].value_counts()
```

[ ]:
```
train['Sex'].value_counts()
```

0.15 make dummies

[ ]:
```
training=pd.get_dummies(train, columns=["Pclass","Sex"], drop_first=True)
training
```

0.16 Standardize the Age and the Fare

[ ]:
```
from sklearn.preprocessing import StandardScaler
train_standard = StandardScaler()
train_copied = training.copy()
train_standard.fit(train_copied[['Age', 'Fare']])
train_std = pd.DataFrame(train_standard.transform(train_copied[['Age',␣
  ↪'Fare']]))
train_std
```

[ ]:
```
training[['Age','Fare'] ] = train_std
training
```

0.17 Since the target variable is survival, let's fit a logistic regression model to predict binary values.

[ ]:
```
from sklearn.linear_model import LogisticRegression

cols = ["Age","Fare","Alone","Pclass_2","Pclass_3","Sex_male"]
X =training[cols]
y = training['Survived']

model = LogisticRegression()
model.fit(X,y)
```

1 Let's check the accuracy of predictions using the trained data.

[ ]:
```
from sklearn.metrics import accuracy_score
train_predicted =model.predict(X)
accuracy_score( train_predicted, y)
```

1.1 Since the results are not very favorable, let's try changing the model.

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
model =␣
  ↪RandomForestClassifier(bootstrap=True,n_estimators=10,criterion='gini',max_depth=None,rando
model.fit(X,y)
```

[ ]:
```
predicted_train2 = model.predict(X)
accuracy_score(predicted_train2 , y)
```

Since the prediction accuracy has improved, let's use this model to predict the test data.

Apply the same preprocessing steps that were performed on the training data to the test data.

```
[ ]: test = pd.read_csv('/kaggle/input/titanic/test.csv')
     test.head()
```

```
[ ]: test.drop(['PassengerId','Name','Cabin','Ticket'], axis=1, inplace=True)
```

```
[ ]: TrainMedian = train['Age'].median()
     TrainMedian
```

```
[ ]: test["Age"].fillna(TrainMedian, inplace=True)
     test["Fare"].fillna(train.Fare.median(), inplace=True)
     test['Alone']=np.where((test["SibSp"]+test["Parch"])>0, 0, 1)
     test.drop(['SibSp', 'Parch'], axis=1, inplace=True)
     testing=pd.get_dummies(test, columns=["Pclass","Sex"], drop_first=True)
     testing
```

```
[ ]: test_copied = testing.copy()
     test_std = train_standard.transform(test_copied[['Age','Fare']])

     testing[['Age','Fare']] = test_std
     testing
```

1.2 Using the model trained on the training data, predict the survival rates for the test data.

```
[ ]: X_test=testing[cols]
     test_predicted = model.predict(X_test)
```

```
[ ]: test_predicted
```

1.3 To submit the predictions, save the results to a file."

```
[ ]: sub = pd.read_csv('../input/titanic/gender_submission.csv')
     sub['Survived'] = list(map(int, test_predicted))
     sub.to_csv('submission.csv', index=False)
```

1.4 How to submit the file

To begin, press the [Run All] button on the screen to execute the code. If it runs properly, a file named ‘submission.csv’ should be generated. Look for the tab on the right side called [Competitions] and find the [Submit] button. Click on it to complete the submission. Let’s check the results. Go back to the competition’s description page and look at the Leaderboard. Your name and score should be displayed!

https://www.kaggle.com/code/wakepon/starter-1