titanic_problem

March 10, 2023

\#

Titanic - Machine Learning from Disaster

0.1 Brief about the problem

I am trying hands on basics of machine learning and found this Titanic problem as one of the best to practice on. This is one the legendary problems on Kaggle platform.

The challenge is about one of the most infamous events in history - Titanic shipwreck. During its first journey on April 15, 1912, the RMS Titanic, which was thought to be invincible, sank after hitting an iceberg. Regrettably, the insufficient number of lifeboats available on board resulted in the loss of 1502 out of 2224 passengers and crew.

Although luck played a role in determining survival, it appears that certain groups had a higher chance of surviving than others. The task at hand is to construct a forecasting model that answers the query, "Which groups of individuals had a greater likelihood of surviving?" by utilizing passenger data such as name, age, gender, socio-economic class, and other relevant factors.

0.2 About the dataset

We have been provided with two datasets, train.csv and test.csv, which contain comparable passenger information such as name, age, gender, socio-economic class, and more.

train.csv dataset contains information about a subset of passengers who were onboard (specifically, 891 individuals) and is crucial as it indicates whether or not they survived, which is commonly referred to as the "ground truth."

test.csv dataset contains similar information except the "ground truth" for each passenger. Using the pattern found in the train.csv dataset, we need to predict whether the other 418 passengers on board found in test.csv dataset survived or not.

0.3 Reference

I have referred from the notebook provided by Alexis Cook in the official competition on Kaggle.

Link - https://www.kaggle.com/code/alexisbcook/titanic-tutorial/notebook

---

0.4 Execution

```
[22]: ## import required libraries
      import numpy as np # linear algebra
      import pandas as pd # data processing
```

0.4.1 load data

Load the data into the workspace to process into a usable format and use for training machine learning model.

```
[2]: train_data = pd.read_csv("./data/train.csv")
     train_data.tail()
```

```
[2]:      PassengerId  Survived  Pclass                                       Name  \
     886          887         0       2                      Montvila, Rev. Juozas
     887          888         1       1               Graham, Miss. Margaret Edith
     888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"
     889          890         1       1                      Behr, Mr. Karl Howell
     890          891         0       3                        Dooley, Mr. Patrick

             Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
     886    male  27.0      0      0      211536  13.00   NaN        S
     887  female  19.0      0      0      112053  30.00   B42        S
     888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
     889    male  26.0      0      0      111369  30.00  C148        C
     890    male  32.0      0      0      370376   7.75   NaN        Q
```

```
[3]: test_data = pd.read_csv("./data/test.csv")
     test_data.tail()
```

```
[3]:      PassengerId  Pclass                          Name     Sex   Age  SibSp  \
     413         1305       3             Spector, Mr. Woolf    male   NaN      0
     414         1306       1  Oliva y Ocana, Dona. Fermina  female  39.0      0
     415         1307       3     Saether, Mr. Simon Sivertsen    male  38.5      0
     416         1308       3              Ware, Mr. Frederick    male   NaN      0
     417         1309       3        Peter, Master. Michael J    male   NaN      1

          Parch            Ticket      Fare Cabin Embarked
```

```
413       0            A.5. 3236    8.0500   NaN        S
414       0              PC 17758  108.9000   C105       C
415       0  SOTON/O.Q. 3101262    7.2500   NaN        S
416       0                359309    8.0500   NaN        S
417       1                  2668   22.3583   NaN        C
```

[4]: `print("Shape Train data {}, Test data {}".format(train_data.shape, test_data.` `↪shape))`

```
Shape Train data (891, 12), Test data (418, 11)
```

0.5 Data Processing

Process the data to get it in a usable format, clean it and replace/remove NaN values if required

[5]:
```
## Drop the useless columns
train_data.drop(['Name','Cabin','Ticket'], axis=1, inplace=True)
test_data.drop(['Name','Cabin','Ticket'], axis=1, inplace=True)
```

[6]:
```
## check if data contains nan values
train_data.isna().sum()
```

[6]:
```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```

[7]: `test_data.isna().sum()`

[7]:
```
PassengerId      0
Pclass           0
Sex              0
Age             86
SibSp            0
Parch            0
Fare             1
Embarked         0
dtype: int64
```

```python
[8]:  ## Fill na for Embarked column using most frequent value
      train_data['Embarked'].fillna(train_data['Embarked'].mode()[0], inplace = True)
      test_data['Embarked'].fillna(test_data['Embarked'].mode()[0], inplace = True)
```

```python
[9]:  ## Fill na for Fare column using median value
      test_data['Fare'].fillna(test_data['Fare'].median(), inplace = True)
```

```python
[10]: ## Fill na for Age column using mean-std, mean+std range
      mean_train_data = train_data["Age"].mean()
      std_train_data = train_data["Age"].std()

      missing_count = train_data["Age"].isna().sum()
      ## generate random numbers in the range of mean-std , mean_std to fill NaNs
      random_gen_age = np.random.randint(mean_train_data - std_train_data,
       ↪mean_train_data + std_train_data, size = missing_count)

      ## fill nan
      train_data_age = train_data["Age"].copy()
      train_data_age[np.isnan(train_data_age)] = random_gen_age
      train_data["Age"] = train_data_age

      ## convert data type to int
      train_data["Age"] = train_data["Age"].astype(int)
```

```python
[11]: ## Fill na for Age column using mean-std, mean+std range
      mean_test_data = test_data["Age"].mean()
      std_test_data = test_data["Age"].std()

      missing_count = test_data["Age"].isnull().sum()
      ## generate random numbers in the range of mean-std , mean_std to fill NaNs
      random_gen_age = np.random.randint(mean_test_data - std_test_data,
       ↪mean_test_data + std_test_data, size = missing_count)

      ## fill nan
      test_data_age = test_data["Age"].copy()#creating a copy for further use
      test_data_age[np.isnan(test_data_age)] = random_gen_age
      test_data["Age"] = test_data_age

      ## convert data type to int
      test_data["Age"] = test_data["Age"].astype(int)
```

```python
[12]: ## Transform Pclass column into categories
      train_data['Pclass'] = train_data.Pclass.astype('category')
      test_data['Pclass'] = test_data.Pclass.astype('category')
```

```
[13]: ## Transform age column into category
      bins = [0,18,50,150]
      labels=['Child','Adult','Senior']

      ## Categories - Child, Adult, Senior
      train_data['Age_'] = pd.cut(train_data['Age'], bins=bins, labels=labels,␣
       ↪right=False)
      train_data.drop('Age', axis = 1, inplace=True)
      test_data['Age_'] = pd.cut(test_data['Age'], bins=bins, labels=labels,␣
       ↪right=False)
      test_data.drop('Age', axis=1, inplace=True)
```

```
[19]: ## prepare data for training and testing
      train_data_X = pd.get_dummies(train_data.drop('Survived', axis=1))
      test_data_X = pd.get_dummies(test_data)
      train_data_X.head(5)
```

```
[19]:    PassengerId  SibSp  Parch     Fare  Pclass_1  Pclass_2  Pclass_3  \
      0            1      1      0   7.2500         0         0         1
      1            2      1      0  71.2833         1         0         0
      2            3      0      0   7.9250         0         0         1
      3            4      1      0  53.1000         1         0         0
      4            5      0      0   8.0500         0         0         1

         Sex_female  Sex_male  Embarked_C  Embarked_Q  Embarked_S  Age__Child  \
      0           0         1           0           0           1           0
      1           1         0           1           0           0           0
      2           1         0           0           0           1           0
      3           1         0           0           0           1           0
      4           0         1           0           0           1           0

         Age__Adult  Age__Senior
      0           1            0
      1           1            0
      2           1            0
      3           1            0
      4           1            0
```

```
[20]: ## final check for nan values
      train_data_X.isna().sum()
```

```
[20]: PassengerId    0
      SibSp          0
      Parch          0
      Fare           0
      Pclass_1       0
      Pclass_2       0
```

```
Pclass_3         0
Sex_female       0
Sex_male         0
Embarked_C       0
Embarked_Q       0
Embarked_S       0
Age__Child       0
Age__Adult       0
Age__Senior      0
dtype: int64
```

[21]: `test_data_X.isnull().sum()`

```
[21]: PassengerId    0
      SibSp          0
      Parch          0
      Fare           0
      Pclass_1       0
      Pclass_2       0
      Pclass_3       0
      Sex_female     0
      Sex_male       0
      Embarked_C     0
      Embarked_Q     0
      Embarked_S     0
      Age__Child     0
      Age__Adult     0
      Age__Senior    0
      dtype: int64
```

0.6   Model Training

I will be training a Random Forest Classifier to fulfill the goal of this problem.

```python
[38]: ## import model library
      from sklearn.ensemble import RandomForestClassifier

      ## label to provide to the model
      y = train_data["Survived"]

      ## set the hyper parameters for the model
      model = RandomForestClassifier(n_estimators=100, max_depth=8, random_state=1)
      model.fit(train_data_X, y)

      ## predict output
```

```
predictions = model.predict(test_data_X)
```

[39]:
```
## check the training accuracy
from sklearn.metrics import accuracy_score

accuracy_train = accuracy_score(y, model.predict(train_data_X))
print("Training accuracy is {} %".format(round(accuracy_train*100, 2)))
```

Training accuracy is 92.03 %

[40]:
```
## format the output for submission
output = pd.DataFrame({'PassengerId': test_data.PassengerId,
                       'Survived': predictions})
## save the output as a csv file
output.to_csv('submission.csv', index=False)
```

0.7 Contribution

While Alexis Cook provided a great introduction about the problem, that is not the best possible solution. In this notebook, I have modified the data processing and code in a manner to produce better output than the reference; **3% better** to be precise.

My contribution includes the following aspects mentioned in this notebook - - Using better feature space : I am using more features, namely SibSp, Parch, Pclass, Sex, Fare, Embarked, and Age. - The additional features i.e. Embarked and Age are converted to categorical type as they are categorical features - Age has been converted from numerical to categorical feature by defining brackets for age groups - NaN values are also handled in Fare, Embarked and Age features - The Categorical features are filled with most frequent value to fill NaN values - The numerical feature Fare is filled with median value to fill NaN values - Using the RandomForestClassifier in a better way i.e. tuning the hyperparameters like max_depth and n_estimators. - I also tried the Support Vector Machine model to compare the accuracy. Random Forest Classifier turns out to be better especially in case of testing dataset.

0.8 Output

Finally, we can get the output generated from the machine learning model. This output is generated as a CSV file and can be submitted in the Kaggle competition to see the testing accuracy.