Titanic LogReg Experiments

By: A. Bounds

Env Setup

Firstly, we need to import our needed packages.

In [ ]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Features

The full list of features can be described in this segment of code, where each individual feature is take from this.

In [ ]:
```
titanic_data = pd.read_csv("titanic_train.csv")
all_features = titanic_data[["PassengerId", "Pclass", "Name", "Sex", \
                             "Age", "SibSp", "Parch", "Ticket", "Fare", "Cabin", \
                             "Embarked"]]
```

Test Runner

Our little bit of code to test each set of features.

In [ ]:
```
def test_feature_set(features, target):
    feature_train, feature_test, target_train, target_test = train_test_split(featu

    # feature_train.reshape(-1, 1)
    # feature_test.reshape(-1, 1)

    model = LogisticRegression()
    model.fit = model.fit(feature_train, target_train)

    preds = model.fit.predict(feature_test)

    print("Confusion Matrix: ")
    print(confusion_matrix(target_test, preds))

    print("\nAccuracy Score: ")
    print(accuracy_score(target_test, preds))
```

Test Cases

Next up, lets build some tests.

Passenger ID

In [ ]:
```
features = titanic_data[["PassengerId"]]
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[168   0]
 [100   0]]

Accuracy Score:
0.6268656716417911
```

P Class

In [ ]:
```
features = titanic_data[["Pclass"]]
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[139  26]
 [ 62  41]]

Accuracy Score:
0.6716417910447762
```

Sex

In [ ]:
```
features = titanic_data[["Sex"]].applymap(lambda x: int(x == "male"))
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[127  29]
 [ 37  75]]

Accuracy Score:
0.753731343283582
```

Age

In [ ]:
```
def int_mapper(x):
    try:
        return int(x)
    except:
        return -1
```

In [ ]:
```
features = titanic_data[["Age"]].applymap(int_mapper, na_action=None)
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[164   0]
 [104   0]]

Accuracy Score:
0.6119402985074627
```

SibSp

In [ ]:
```
features = titanic_data[["SibSp"]]
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[162   0]
 [106   0]]

Accuracy Score:
0.6044776119402985
```

Parch

In [ ]:
```
features = titanic_data[["Parch"]]
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[177   2]
 [ 87   2]]

Accuracy Score:
0.667910447761194
```

Fare

In [ ]:
```
features = titanic_data[["Fare"]]
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[148  11]
 [ 84  25]]

Accuracy Score:
0.6455223880597015
```

Cabin

In [ ]:
```python
def str_to_int(x):
    try:
        return sum([ord(c) for c in x])
    except:
        return -1
```

In [ ]:
```python
features = titanic_data[["Cabin"]].applymap(str_to_int, na_action=None)
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[142  25]
 [ 60  41]]

Accuracy Score:
0.6828358208955224
```

Embarked

In [ ]:
```python
features = titanic_data[["Embarked"]].applymap(str_to_int, na_action=None)
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[155  17]
 [ 67  29]]

Accuracy Score:
0.6865671641791045
```

Remarks on Excluded Data

Some of the data, such as the passengers name or ticket, were excluded from these tests. It was determined it would be pointless to check these data values. For example, what use would a person name be, used all on it's own, to determine if a person had a likelihood of survival? There isn't any, so it wasn't tested.

Results

The best estimation, by the tests conducted here, lies with the Sex of the passenger.

In [ ]:
```
features = titanic_data[["Sex"]].applymap(lambda x: int(x == "male"))
target = titanic_data.Survived

test_feature_set(features, target)
```

```
Confusion Matrix:
[[138  26]
 [ 37  67]]

Accuracy Score:
0.7649253731343284
```

As seen above, the accuracy score is much higher than most other results (~ `0.65`) and demonstrates a correlation between the Sex of the passenger and their survival of the sinking of the titanic.

Conclusions

Firstly, I would like to make a comment on the usage of a single point of data to determine a correlation. It doesn't make much sense, and I'm sure that if I tested this dataset with more than just the Sex of the passenger, there would be an even stronger correlation to be found. As it stands, however, the results of this set of testing are very clear.

The Sex of the passenger makes sense in this instance, especially if you reflect back to the time period in which the ship sunk. Having the women and children get off the ship first would have been a priority for those onboard.

Overall, there isn't anything too interesting in this dataset. It lines up with my own conclusions and doesn't take me much to look understand why.