regression_titanic

July 8, 2021

1 Regression on the Titanic dataset

- Link to dataset

[1]:
```python
path= "./data/"
csv_train= "train.csv"
csv_test= "test.csv"
import pandas as pd
train= pd.read_csv(path+csv_train)
test= pd.read_csv(path+csv_test)
data= train.copy()
```

1.1 EDA

[2]:
```python
data.columns
```

[2]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

[3]:
```python
data.isna().sum()
```

[3]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[4]:
```python
data.shape
```

[4]: (891, 12)

[5]:
```
data[data.Survived==1].Cabin.value_counts()
```

[5]:
```
B96 B98    4
E101       3
F33        3
D17        2
B28        2
          ..
C62 C64    1
E12        1
B4         1
C50        1
C99        1
Name: Cabin, Length: 101, dtype: int64
```

[6]:
```
data[data.Embarked.isna()].Survived
```

[6]:
```
61     1
829    1
Name: Survived, dtype: int64
```

[7]:
```
data.Embarked.value_counts()
```

[7]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

[8]:
```
data[data.Embarked=="S"].Survived.value_counts()
```

[8]:
```
0    427
1    217
Name: Survived, dtype: int64
```

[9]:
```
data.Fare.value_counts()
```

[9]:
```
8.0500     43
13.0000    42
7.8958     38
7.7500     34
26.0000    31
           ..
50.4958     1
13.8583     1
8.4583      1
7.7250      1
```

```
7.5208      1
Name: Fare, Length: 248, dtype: int64
```

[10]:
```python
data.Pclass.value_counts()
```

[10]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

[11]:
```python
data.groupby("Pclass")[["Survived"]].agg({"Survived": "sum"})
```

[11]:
```
        Survived
Pclass
1            136
2             87
3            119
```

[12]:
```python
df= pd.pivot_table(data=data,index="Pclass",values="Survived",columns="Sex")
df
```

[12]:
```
Sex       female      male
Pclass
1       0.968085  0.368852
2       0.921053  0.157407
3       0.500000  0.135447
```

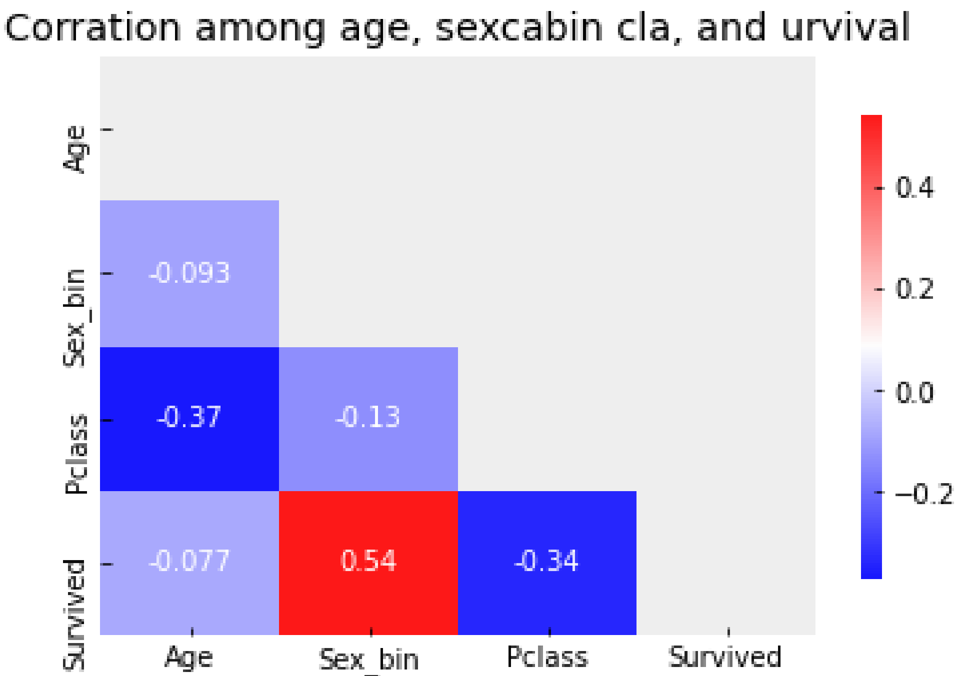
[13]:
```python
data["Sex_bin"]= pd.factorize(data.Sex)[0]
# data.Sex_bin
```

[14]:
```python
# correlation
corr_mat= data[["Age","Sex_bin","Pclass","Survived"]].corr()
corr_mat
```

[14]:
```
               Age   Sex_bin    Pclass  Survived
Age       1.000000 -0.093254 -0.369226 -0.077221
Sex_bin  -0.093254  1.000000 -0.131900  0.543351
Pclass   -0.369226 -0.131900  1.000000 -0.338481
Survived -0.077221  0.543351 -0.338481  1.000000
```

[15]:
```python
import numpy as np
mask= np.triu(np.ones_like(corr_mat,dtype=bool))

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use("bmh") # seaborn-poster
# sns.set_theme(palette="bright",style="darkgrid")
```

```
sns.heatmap(corr_mat,annot=True,mask=mask,cmap=sns.color_palette("bwr",␣
↪as_cmap=True),alpha=.9,cbar_kws={"shrink":.8})
plt.title("Corration among age, sexcabin cla, and urvival")#
# from matplotlib import rcParams rcParams["axes.titlepad"]= 20
plt.show()
```



1.2 Preprocessing

- Fill in empty cells of `Age`, `Embarked`
- Drop non-useful columns: `Ticket`, `Cabin`, `Name`

[16]:
```
data.groupby("Sex")["Age"].mean(), data.groupby("Sex")["Age"].median()
```

[16]:
```
(Sex
 female    27.915709
 male      30.726645
 Name: Age, dtype: float64,
 Sex
 female    27.0
 male      29.0
 Name: Age, dtype: float64)
```

[17]:
```
data.Age.median(), data.Age.mean()
```

[17]: (28.0, 29.69911764705882)

[18]:
```
# - Fill in empty cells of `Age`
# transform()
# https://stackoverflow.com/questions/40957932/transform-vs-aggregate-in-pandas
data.Age.fillna(data.groupby("Sex")["Age"].transform("median"),inplace=True)
```

[19]:
```
# - Fill in empty cells of `Embarked` with the last value before NaN
data.Embarked.fillna(method="pad",inplace=True)
```

[20]:
```
# Drop non-useful columns: `Ticket`, `Cabin`
data.drop(["Ticket","Cabin","Name","Sex"],axis=1,inplace=True)
```

1.3 Feature Engineering

- Add features: Age_bin, Fare_bin, Solo

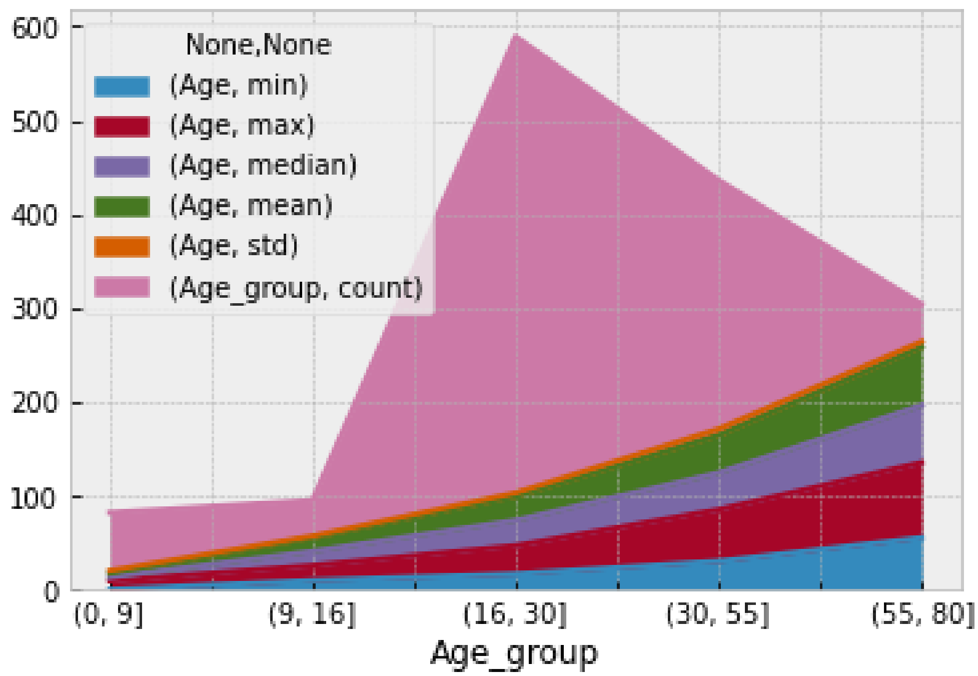
[21]:
```
data.Age.max()
```

[21]: 80.0

[22]:
```
data["Age_group"]= pd.cut(data.Age,bins=[0,9,16,30,55,80]) #
df_agegroup= data[["Age_group","Survived","Age"]].groupby("Age_group").
↪agg(dict(Age=["min","max","median","mean","std"], Age_group=["count"]))
```

[23]:
```
print(df_agegroup)
df_agegroup.plot(kind="area")#,x="Age_group",y="Age"
plt.style.use("Solarize_Light2")

plt.show()
```

```
              Age                                           Age_group
              min   max median       mean       std     count
Age_group
(0, 9]       0.42   9.0    4.0   4.083387  2.834747        62
(9, 16]     10.00  16.0   15.0  14.407895  1.951629        38
(16, 30]    17.00  30.0   27.0  25.387860  3.939789       486
(30, 55]    30.50  55.0   39.0  39.996226  6.789725       265
(55, 80]    55.50  80.0   61.0  62.350000  5.619563        40
```



[24]:
```
data.Age_group.value_counts()
```

[24]:
```
(16, 30]    486
(30, 55]    265
(0, 9]       62
(55, 80]     40
(9, 16]      38
Name: Age_group, dtype: int64
```

[25]:
```
df_age= data.groupby("Age_group")["Age"].
 ↪agg(["min","max","median","mean","std","size"])
print(df_age)
```

```
             min   max  median       mean       std  size
Age_group
(0, 9]      0.42   9.0     4.0   4.083387  2.834747    62
(9, 16]    10.00  16.0    15.0  14.407895  1.951629    38
(16, 30]   17.00  30.0    27.0  25.387860  3.939789   486
(30, 55]   30.50  55.0    39.0  39.996226  6.789725   265
(55, 80]   55.50  80.0    61.0  62.350000  5.619563    40
```

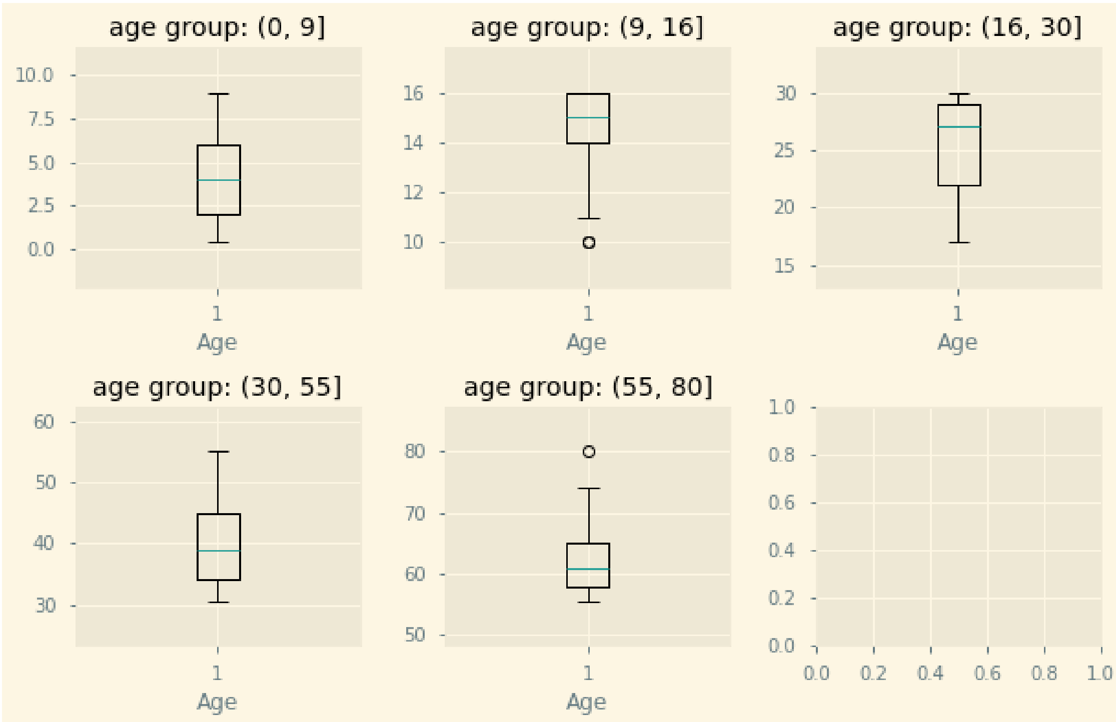
[26]:
```
data["Age_bin"]= pd.Categorical(data.Age_group).codes
data.head(1)
```

[26]:
```
   PassengerId  Survived  Pclass   Age  SibSp  Parch  Fare Embarked  Sex_bin  \
0            1         0       3  22.0      1      0  7.25        S        0

  Age_group  Age_bin
0  (16, 30]        2
```

[27]:
```
data[["Age_group","Age_bin"]].value_counts()
```

[27]:
```
Age_group  Age_bin
(16, 30]   2          486
(30, 55]   3          265
(0, 9]     0           62
(55, 80]   4           40
(9, 16]    1           38
dtype: int64
```

[28]:
```
plt.style.use("seaborn-notebook")
# data.groupby("Age_bin")[["Age"]].
↪plot(kind="box",subplots=True,grid=True,sharex=True,sharey=False,title="Boxplot␣
↪for Each Age Group")
grouped= data.groupby("Age_group")[["Age"]]
ncols= int(grouped.ngroups/2)+(grouped.ngroups%2) #round up
fig,ax= plt.
↪subplots(figsize=(10,6),nrows=2,ncols=ncols,gridspec_kw=dict(hspace=.
↪5,wspace=0.3))
groups= zip(grouped.groups.keys(),ax.flatten())
for i,(key,axis) in enumerate(groups):
    axis.boxplot(grouped.get_group(key))
    axis.set_xlabel("Age",fontsize=12)
    axis.set_title("age group: %s"%str(key),fontsize=14)
    axis.margins(x=.1,y=.3)

plt.show()
```



[29]:
```
data.insert(7,"Embarked_bin",pd.Categorical(data.Embarked).codes)
```

[30]:
```
data["Fare_group"]= pd.cut(data.Fare,bins=[0,8,15,31,513])
data.insert(6,"Fare_bin",pd.Categorical(data.Fare_group).codes)
```

[31]:
```
data.insert(4,"In_company",((data.SibSp+data.Parch)>0).astype(int))
```

[32]:
```
data.head(3)
```

[32]:
```
   PassengerId  Survived  Pclass   Age  In_company  SibSp  Parch  Fare_bin  \
0            1         0       3  22.0           1      1      0         0
1            2         1       1  38.0           1      1      0         3
2            3         1       3  26.0           0      0      0         0

      Fare  Embarked_bin Embarked  Sex_bin Age_group  Age_bin Fare_group
0   7.2500             2        S        0  (16, 30]        2     (0, 8]
1  71.2833             0        C        1  (30, 55]        3  (31, 513]
2   7.9250             2        S        1  (16, 30]        2     (0, 8]
```

[33]:
```
data.Pclass.value_counts()/data.shape[0]
```

[33]:
```
3    0.551066
1    0.242424
2    0.206510
```

```
Name: Pclass, dtype: float64
```

[34]:
```
data.columns
```

[34]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'In_company', 'SibSp',
       'Parch', 'Fare_bin', 'Fare', 'Embarked_bin', 'Embarked', 'Sex_bin',
       'Age_group', 'Age_bin', 'Fare_group'],
      dtype='object')
```

[35]:
```
# data.dtypes
```

[36]:
```
cols= ['Pclass', 'In_company','Embarked_bin', 'Fare_bin',␣
 ↪'Sex_bin','Age_bin',"Fare","SibSp","Age"]
X= data[cols].values
y= data.Survived.values

import statsmodels.api as sm
X= sm.add_constant(X)

model_ols= sm.OLS(y,X)
res= model_ols.fit()
print(res.summary(xname=["Intercept"]+cols,yname="Survived"))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               Survived   R-squared:                       0.401
Model:                            OLS   Adj. R-squared:                  0.395
Method:                 Least Squares   F-statistic:                     65.64
Date:                Thu, 08 Jul 2021   Prob (F-statistic):           4.41e-92
Time:                        06:52:35   Log-Likelihood:                -393.35
No. Observations:                 891   AIC:                             806.7
Df Residuals:                     881   BIC:                             854.6
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept        0.7864      0.091      8.609      0.000       0.607       0.966
Pclass          -0.1537      0.024     -6.471      0.000      -0.200      -0.107
In_company       0.0525      0.037      1.421      0.156      -0.020       0.125
Embarked_bin    -0.0330      0.017     -1.978      0.048      -0.066      -0.000
Fare_bin         0.0163      0.019      0.852      0.394      -0.021       0.054
Sex_bin          0.4859      0.028     17.134      0.000       0.430       0.542
Age_bin         -0.0405      0.036     -1.126      0.260      -0.111       0.030
Fare             0.0002      0.000      0.454      0.650      -0.001       0.001
SibSp           -0.0681      0.016     -4.366      0.000      -0.099      -0.038
Age             -0.0031      0.002     -1.315      0.189      -0.008       0.002
==============================================================================
```

```
Omnibus:                       33.711   Durbin-Watson:                   1.946
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               36.640
Skew:                           0.493   Prob(JB):                     1.11e-08
Kurtosis:                       3.128   Cond. No.                         464.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[37]:

```python
R_sqrd= []
from sklearn.model_selection import train_test_split
for i in range(100):
    X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=.3)
    model= sm.OLS(y_train,X_train)
    res= model.fit()
    pred= res.predict(X_test)
    R2= 1 - sum((pred-y_test)**2)/sum((y_test-y_test.mean())**2)
    R_sqrd.append(R2)
print("R_squared",np.mean(R_sqrd))
```

```
R_squared 0.3737962337998663
```

[38]:

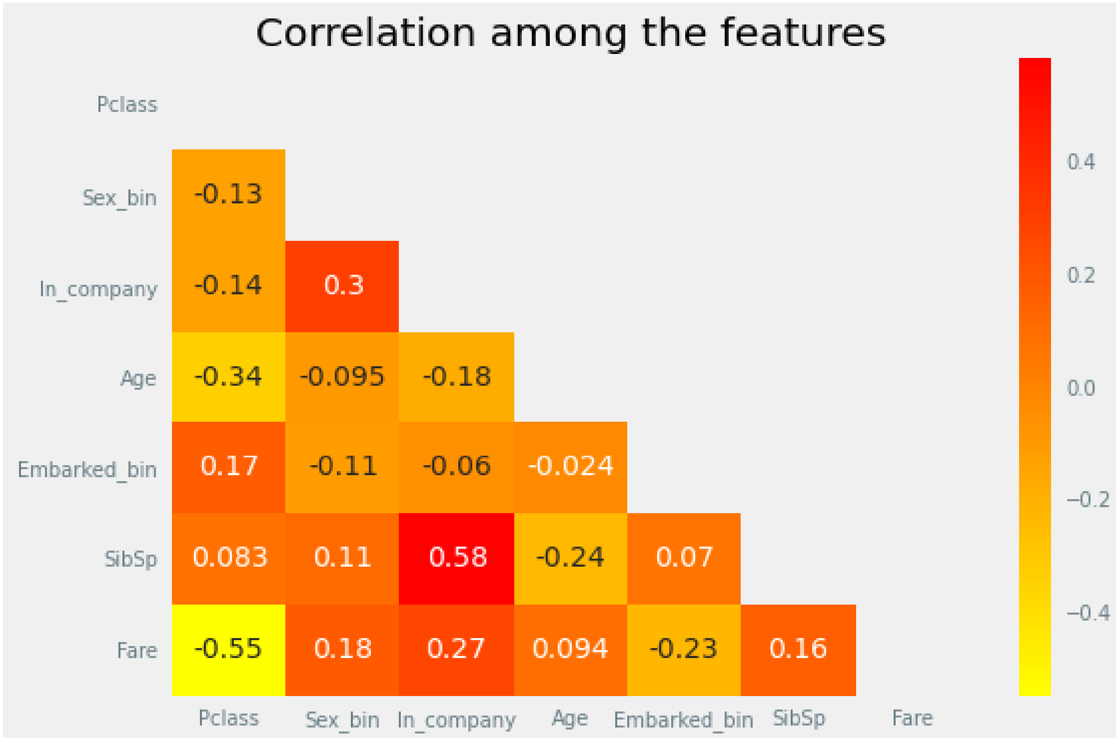
```python
set(cols).difference(set(["Age_bin","Fare_bin"]))
```

[38]:

```
{'Age', 'Embarked_bin', 'Fare', 'In_company', 'Pclass', 'Sex_bin', 'SibSp'}
```

[39]:

```python
plt.style.use("fivethirtyeight")
corr_mat= data[set(cols).difference(set(["Age_bin","Fare_bin"]))].corr()
mask= np.triu(np.ones_like(corr_mat,dtype=bool))
sns.heatmap(corr_mat,mask=mask,cbar=True,annot=True,cmap="autumn_r")
plt.title("Correlation among the features")
plt.show()
```



1.4 Next Step: Look at outliers in the data