wqtb3v17y

April 29, 2024

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

[ ]:
```
test_df = pd.read_csv("test.csv")
train_df=pd.read_csv("train.csv")
```

[ ]:
```
import statistics
```

[ ]:
```
u=statistics.mean(train_df['Age'])
```

[ ]:
```
print(u)
```

```
nan
```

[ ]:
```
mean = sum(train_df['Age']) / len(train_df['Age'])
```

[ ]:
```
print(mean)
```

```
nan
```

[ ]:
```
data = [train_df, test_df]

for dataset in data:
    mean = train_df["Age"].mean()
    std = test_df["Age"].std()
    is_null = dataset["Age"].isnull().sum()
    # compute random numbers between the mean, std and is_null
    rand_age = np.random.randint(mean - std, mean + std, size = is_null)
    # fill NaN values in Age column with random values generated
    age_slice = dataset["Age"].copy()
    age_slice[np.isnan(age_slice)] = rand_age
    dataset["Age"] = age_slice
    dataset["Age"] = train_df["Age"].astype(int)
train_df["Age"].isnull().sum()
```

[ ]:
```
0
```

[ ]:
```
std_dev=statistics.stdev(train_df['Age'])
print(std_dev)
```

```
13.49383039791902
```

Z-Score

[ ]:
```
z=(train_df['PassengerId'],(train_df['Age']-mean)/std_dev)
print(z)
```

```
(0        1
1        2
2        3
3        4
4        5
      …
886    887
887    888
888    889
889    890
890    891
Name: PassengerId, Length: 891, dtype: int64, 0     -0.554104
1      0.631623
2     -0.257673
3      0.409299
4      0.409299
         …
886   -0.183565
887   -0.776428
888    0.483407
889   -0.257673
890    0.186975
Name: Age, Length: 891, dtype: float64)
```

[ ]:
```
m=max(train_df['Age'])
```

[ ]:
```
mi=train_df['Survived'].min()
```

[ ]:
```
mi=train_df['Age'].min()
print(mi)
```

```
0
```

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

[ ]:
```
       PassengerId    Survived      Pclass          Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.476992    0.523008
```

```
std     257.353842    0.486592    0.836071   13.493830    1.102743
min       1.000000    0.000000    1.000000    0.000000    0.000000
25%     223.500000    0.000000    2.000000   21.000000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   37.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000
```

```
            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

Mean Normalization

[ ]:
```
mean_normalization=(train_df['Age']-mean)/m-mi
print(mean_normalization)
```

```
0     -0.093462
1      0.106538
2     -0.043462
3      0.069038
4      0.069038
         …
886   -0.030962
887   -0.130962
888    0.081538
889   -0.043462
890    0.031538
Name: Age, Length: 891, dtype: float64
```

[ ]:
```
abs(train_df['Fare'])
```

[ ]:
```
0       7.2500
1      71.2833
2       7.9250
3      53.1000
4       8.0500
         …
886    13.0000
887    30.0000
888    23.4500
889    30.0000
890     7.7500
```

```
Name: Fare, Length: 891, dtype: float64
```

Min-max

[ ]:
```
from sklearn.preprocessing import minmax_scale
```

[ ]:
```
print(minmax_scale(train_df['Age']))
```

```
[0.275  0.475  0.325  0.4375 0.4375 0.2375 0.675  0.025  0.3375 0.175
 0.05   0.725  0.25   0.4875 0.175  0.6875 0.025  0.2375 0.3875 0.3125
 0.4375 0.425  0.1875 0.35   0.1    0.475  0.2    0.2375 0.3875 0.275
 0.5    0.2875 0.3125 0.825  0.35   0.525  0.4125 0.2625 0.225  0.175
 0.5    0.3375 0.3    0.0375 0.2375 0.3875 0.1875 0.5    0.4    0.225
 0.0875 0.2625 0.6125 0.3625 0.8125 0.4875 0.2625 0.35   0.0625 0.1375
 0.275  0.475  0.5625 0.05   0.2    0.45   0.3625 0.2375 0.2125 0.325
 0.4    0.2    0.2625 0.325  0.4    0.3125 0.475  0.225  0.     0.375
 0.275  0.3625 0.5125 0.35   0.2125 0.4125 0.2    0.4125 0.2875 0.3
 0.3625 0.25   0.575  0.325  0.7375 0.3    0.8875 0.2875 0.425  0.425
 0.35   0.5    0.2625 0.4125 0.4625 0.35   0.2625 0.225  0.475  0.5
 0.5875 0.175  0.275  0.25   0.2125 0.2625 0.875  0.3625 0.3    0.025
 0.2625 0.4    0.4    0.4    0.675  0.15   0.425  0.3    0.35   0.5625
 0.4125 0.25   0.5875 0.3625 0.3125 0.2875 0.2375 0.4625 0.2    0.3
 0.275  0.275  0.3    0.2375 0.225  0.2375 0.3375 0.1125 0.45   0.525
 0.6375 0.275  0.6875 0.5    0.4375 0.6375 0.2    0.375  0.2375 0.4625
 0.55   0.5    0.325  0.2125 0.0125 0.1125 0.2875 0.5625 0.2125 0.35
 0.7625 0.05   0.0125 0.2625 0.7    0.225  0.2375 0.625  0.375  0.45
 0.25   0.525  0.1125 0.0125 0.05   0.45   0.2375 0.5625 0.5    0.45
 0.4    0.2375 0.2375 0.0375 0.55   0.725  0.3    0.525  0.2375 0.3
 0.35   0.325  0.425  0.5625 0.225  0.025  0.4    0.325  0.2    0.5
 0.3    0.4375 0.275  0.375  0.4    0.3875 0.3375 0.525  0.4    0.375
 0.2    0.3375 0.6375 0.35   0.475  0.275  0.2375 0.25   0.225  0.4375
 0.4375 0.3625 0.7375 0.0625 0.3    0.2125 0.55   0.1    0.2375 0.4125
 0.325  0.525  0.3625 0.275  0.375  0.55   0.3125 0.3    0.4625 0.675
 0.525  0.3625 0.775  0.375  0.5125 0.3625 0.3    0.375  0.4375 0.625
 0.3625 0.0375 0.65   0.5    0.2625 0.45   0.2    0.3125 0.725  0.4375
 0.4125 0.3125 0.5125 0.4625 0.3625 0.7875 0.5625 0.425  0.0875 0.4375
 0.8125 0.35   0.2    0.2375 0.2    0.4125 0.375  0.275  0.525  0.275
 0.325  0.2375 0.45   0.3    0.3    0.425  0.2875 0.025  0.2875 0.625
 0.5125 0.2    0.2375 0.425  0.3125 0.     0.4125 0.2125 0.375  0.375
 0.3    0.225  0.325  0.35   0.5375 0.325  0.3    0.675  0.3875 0.5
 0.275  0.3375 0.375  0.275  0.5    0.45   0.7625 0.45   0.3875 0.2
 0.4875 0.5625 0.475  0.2    0.4875 0.2125 0.3625 0.5125 0.5625 0.5625
 0.025  0.3    0.35   0.3125 0.45   0.3    0.5    0.3125 0.0375 0.525
 0.2875 0.25   0.1875 0.3125 0.475  0.35   0.275  0.475  0.3625 0.525
 0.5    0.3625 0.5625 0.4375 0.225  0.375  0.75   0.3125 0.3    0.3
 0.3125 0.225  0.2375 0.275  0.0375 0.2625 0.275  0.3375 0.25   0.2375
 0.525  0.0125 0.4    0.4375 0.5125 0.225  0.0125 0.45   0.4625 0.2125
 0.45   0.2625 0.35   0.2875 0.3    0.275  0.3875 0.575  0.2875 0.35
```

0.4875 0.325 0.2625 0.35 0.25 0.425 0.6375 0.0375 0.2625 0.2
0.275 0.375 0.4125 0.3875 0.55 0.225 0.425 0.225 0.375 0.125
0.5 0.2625 0.3625 0.35 0.225 0.2875 0.35 0.2375 0.5 0.4
0.35 0.1875 0.525 0.2125 0.625 0.175 0.2625 0.3 0.8 0.3875
0.5625 0.25 0.3125 0.35 0.4625 0.05 0.1625 0.425 0.0625 0.65
0.45 0.425 0.375 0.6125 0.5 0.3625 0.8125 0.4125 0.625 0.2875
0.6 0.425 0.5875 0.6 0.2 0.475 0.425 0.7 0.525 0.
0.325 0.475 0.4125 0.2875 0.275 0.3875 0.425 0.3625 0.275 0.025
0.1125 0.1875 0.625 0.7875 0.3125 0.3625 0.4375 0.725 0.375 0.1125
0.35 0.2625 0.6875 0.8875 0.2625 0.4 0.675 0.3875 0.3125 0.3
0.2125 0.2625 0.2875 0.4625 0.2 0.225 0.4125 0.3875 0.35 0.325
0.3625 0.425 0.45 0.675 0.3 0.5875 0.425 0.2 0.45 0.4
0.375 0.275 0.4875 0.55 0.525 0.5 0.625 0.35 0.4875 0.2875
0.025 0.3375 0.2125 0.4375 0.375 0.0875 0.5625 0.375 0.2375 0.275
0.45 0.1125 0.1375 0.4 0.625 0.8 0.2375 0.2 0.4125 0.1
0.2125 0.3375 0.475 0.275 0.275 0.775 0.6 0.3375 0.4875 0.45
0.3875 0.5 0.35 0.2875 0.4625 0.3 0.2375 0.3625 0.525 0.4
0.775 0.6625 0.45 0.225 0.2 0.2375 0.425 0.4875 0.2625 0.4
0.3125 0.4875 0.675 0.45 0.25 0.225 0.5875 0.75 0.275 0.35
0.4375 0.65 0.5875 0.3625 0.4625 0.45 0.45 0.6125 0.3625 0.6125
0.3 0.4875 0.4625 0.55 0.4375 0.45 0.375 0.3375 0.275 0.5
0.4875 0.4625 0.4125 0.2375 0.4375 0.3 0.425 0.325 0.05 0.325
0.3375 0.525 0.25 0.2625 0.2625 0.7625 0.7125 0.2625 0.325 0.2875
1. 0.6375 0.4 0.3125 0.1125 0.35 0.4 0.3875 0.5125 0.4375
0.25 0.3 0.025 0.4375 0. 0.6 0.2375 0.7 0.2375 0.2875
0.4625 0.225 0.2625 0.25 0.225 0.3 0.25 0.4 0.2875 0.725
0.625 0.5 0.5875 0.45 0.25 0.4 0.3125 0.45 0.5375 0.3875
0.5 0.3875 0.875 0.3875 0.2875 0.225 0.3 0.225 0.5375 0.45
0.4625 0.3375 0.25 0.175 0.75 0.3125 0.175 0.2375 0.225 0.1875
0.3875 0.05 0.3875 0.3125 0.75 0.65 0.55 0.35 0.6125 0.525
0.225 0.4375 0.225 0.3125 0.325 0.4875 0.5625 0.525 0.275 0.275
0.3 0.45 0.6 0.3625 0.65 0.2375 0.475 0.3375 0.4125 0.4125
0.075 0.2125 0.425 0.625 0.3375 0.25 0.375 0.325 0.3125 0.3125
0.3625 0.1375 0.2875 0.2875 0.2875 0.35 0.6 0.4375 0.375 0.3375
0.2 0.45 0.2625 0.3 0.3875 0.875 0.2 0.375 0.2375 0.3875
0.05 0.075 0.4125 0.2875 0.6 0. 0.35 0.225 0.425 0.4125
0.425 0.5125 0.25 0.45 0.2 0.6375 0.225 0.375 0.35 0.4
0.3 0.6 0.7125 0.35 0.675 0.225 0.3375 0.0625 0.275 0.5375
0.1625 0.2125 0.3625 0.4875 0.3125 0.3125 0.225 0.1 0.0125 0.575
0.3625 0.2 0.2375 0.425 0.3125 0.4875 0.6125 0.3875 0.375 0.375
0.425 0.3875 0.1375 0. 0.3375 0.3875 0.4875 0.225 0.4875 0.4125
0.325 0.4875 0.4375 0.075 0.375 0.3375 0.2875 0.3875 0.5375 0.125
0.65 0.3375 0.475 0.3375 0.025 0.4375 0.5 0.0125 0.35 0.775
0.1875 0. 0.2875 0.2875 0.225 0.4875 0.2625 0.3125 0.4 0.4875
0.25 0.2 0.375 0.425 0.2125 0.525 0.525 0.4375 0.35 0.4625
0.05 0.925 0.1125 0.2 0.55 0.225 0.5625 0.6375 0.3 0.3875
0.5125 0.2625 0.6 0.25 0.3 0.525 0.3375 0.3875 0.2125 0.05
0.325 0.5875 0.4125 0.5875 0.35 0.1875 0.25 0.2375 0.2875 0.7

```
0.3125 0.4125 0.275  0.35   0.3125 0.4875 0.3375 0.2375 0.45   0.325
0.4   ]
```

[ ]:
```
x=(train_df['Fare']-min(train_df['Fare']))/
  ↪(max(train_df['Fare'])-min(train_df['Fare']))
```

Absolute Max

[ ]:
```
max_abs=train_df['Fare']/abs(train_df['Fare'].max())
print(max_abs)
```

```
0      0.014151
1      0.139136
2      0.015469
3      0.103644
4      0.015713
         …
886    0.025374
887    0.058556
888    0.045771
889    0.058556
890    0.015127
Name: Fare, Length: 891, dtype: float64
```

Robust Scaling

[ ]:
```
q1=np.percentile((train_df['Fare']),25)
q3=np.percentile((train_df['Fare']),75)
```

[ ]:
```
print(q1,q3)
```

```
7.9104 31.0
```

[ ]:
```
r=(train_df['Fare']-q1)/(q3-q1)
print(r)
```

```
0     -0.028602
1      2.744651
2      0.000632
3      1.957141
4      0.006046
         …
886    0.220428
887    0.956690
888    0.673013
889    0.956690
890   -0.006947
Name: Fare, Length: 891, dtype: float64
```