T4.

```
#0
assign step:
cluster (3, 3):
(3, 3)
(8, 8)
(6, 6)
(7, 7)
cluster (2, 2):
(1, 2)
(2, 2)
cluster (-3, -3):
(-3, -3)
(-2, -4)
(-7, -7)
update step:
[(6.0, 6.0), (1.5, 2.0), (-4.0, -4.666666666666667)]
#1
assign step:
cluster (6.0, 6.0):
(8, 8)
(6, 6)
(7, 7)
cluster (1.5, 2.0):
(1, 2)
(3, 3)
(2, 2)
cluster (-4.0, -4.666666666666667):
(-3, -3)
(-2, -4)
(-7, -7)
update step:
[(7.0, 7.0), (2.0, 2.3333333333333335), (-4.0, -4.666666666666667)]
```

```
#2
assign step:
cluster (7.0, 7.0):
(8, 8)
(6, 6)
(7, 7)
cluster (2.0, 2.3333333333333335):
(1, 2)
(3, 3)
(2, 2)
cluster (-4.0, -4.666666666666667):
(-3, -3)
(-2, -4)
(-7, -7)
update step:
[(7.0, 7.0), (2.0, 2.3333333333333335), (-4.0, -4.666666666666667)]
```

T5.

```
#0
assign step:
cluster (-7, -7):
(-7, -7)
cluster (2, 2):
(1, 2)
(3, 3)
(2, 2)
(8, 8)
(6, 6)
(7, 7)
cluster (-3, -3):
(-3, -3)
(-2, -4)
update step:
[(-7.0, -7.0), (4.5, 4.666666666666667), (-2.5, -3.5)]
#1
assign step:
cluster (-7.0, -7.0):
(-7, -7)
cluster (4.5, 4.666666666666667):
(1, 2)
(3, 3)
(2, 2)
(8, 8)
(6, 6)
(7, 7)
cluster (-2.5, -3.5):
(-3, -3)
(-2, -4)
update step:
[(-7.0, -7.0), (4.5, 4.666666666666667), (-2.5, -3.5)]
```

T6. From Elbow-method :

Compare the starting points from T4. and T5. we can assume that the starting points from T4 is the better starting points.

Compute the fraction of explained variance then the starting points that give more fraction of explained variance is the better one.

OT1. The best K for this question is 4.

we can use Elbow-method and choose the minimal K that explains at least 95% of the all-data variance.

T7.

```
train["Age"] = train["Age"].fillna(train["Age"].median())
print("median of Age:"+str(train["Age"].median()))
```

```
median of Age:28.0
```

T8.

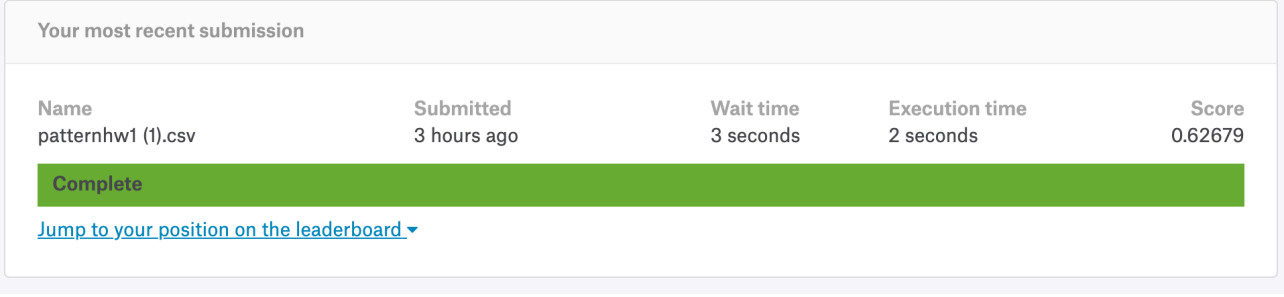
```
train.loc[train["Embarked"] == "S", "Embarked"] = 0
train.loc[train["Embarked"] == "C", "Embarked"] = 1
train.loc[train["Embarked"] == "Q", "Embarked"] = 2
print("mode of Embarked: "+str(train["Embarked"].mode()[0]))
```

```
mode of Embarked: 0
```

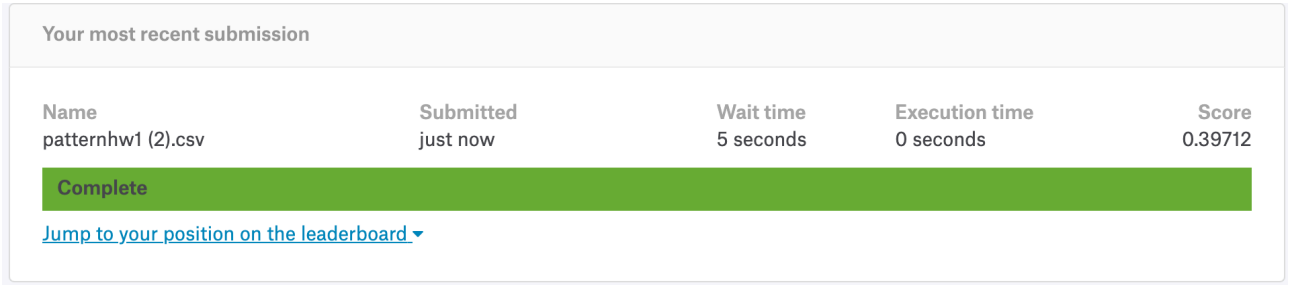
```
train.loc[train["Sex"] == "male", "Sex"] = 0
train.loc[train["Sex"] == "female", "Sex"] = 1
print("mode of Sex: "+str(train["Sex"].mode()[0]))
```

```
mode of Sex: 0
```

T9. & T10. code as submitted in courseville



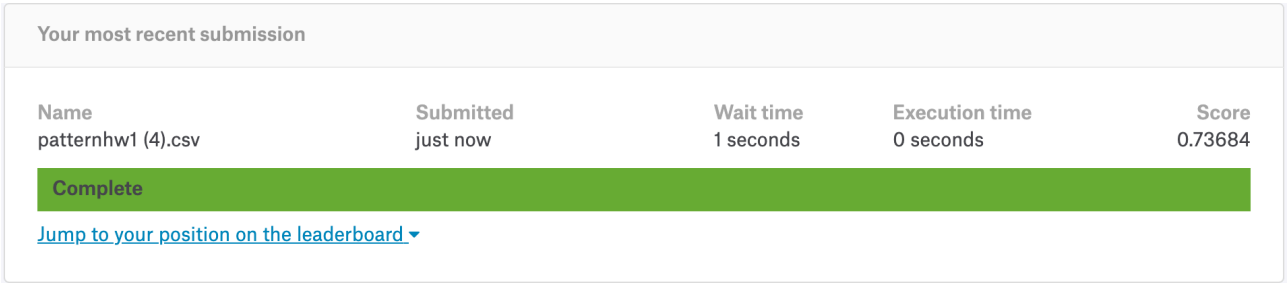
OT2. Linear regression will give the value of y (Passenger is survived or not.) but we don't know if y is more than 1 or y is less than 0, what is the meaning of it so we need to find some value(arithmetic mean) that will classify our answer in 2 different ways.



The result is worse than logistic regression. I think the reason is arithmetic mean is not the good threshold in this case.

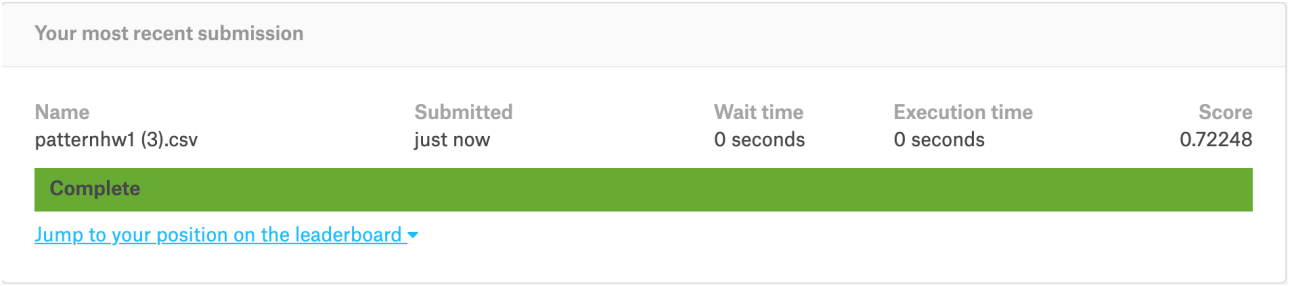
In summary, Yes/No question is more comfortable with logistic regression because the value that we will get is [0,1].

OT3.



The result is better than every method. I think because with gradient descent method, we must try various learning rate and the number of iteration.

OT4. Using Column PClass as a higher order feature make the model more accurate.



I think because the passengers in high class will get some help first (Capitalism) so we can use this column as a higher order feature.

OT5. Your model will have less accuracy because with 891 rows but your column is 2 that make you don't enough features to create the model with high accuracy.