Homework 4 report 2016312761 여혁수

1-(a)

```
from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark import SparkContext
import pandas as pd

spark = SparkSession.builder.appName("Titanic-Dataset").config('spark.driver.memory','15g').getOrCreate()

df = spark.read.option("header", True).csv('train.csv')

df.printSchema()
```

```
root
 |-- PassengerId: string (nullable = true)
 |-- Survived: string (nullable = true)
 |-- Pclass: string (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- SibSp: string (nullable = true)
 |-- Parch: string (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: string (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
```

1-(b)

```
from pyspark.sql.functions import when, count, col
df.select([count(when(isnull(c), c)).alias(c) for c in df.columns]).show()

columns_to_drop = ['Ticket', 'Cabin']
df = df.drop(*columns_to_drop)
```

```
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|PassengerId|Survived|Pclass|Name|Sex|Age|SibSp|Parch|Ticket|Fare|Cabin|Embarked|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|          0|       0|     0|   0|  0|177|    0|    0|     0|   0|  687|       2|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
```

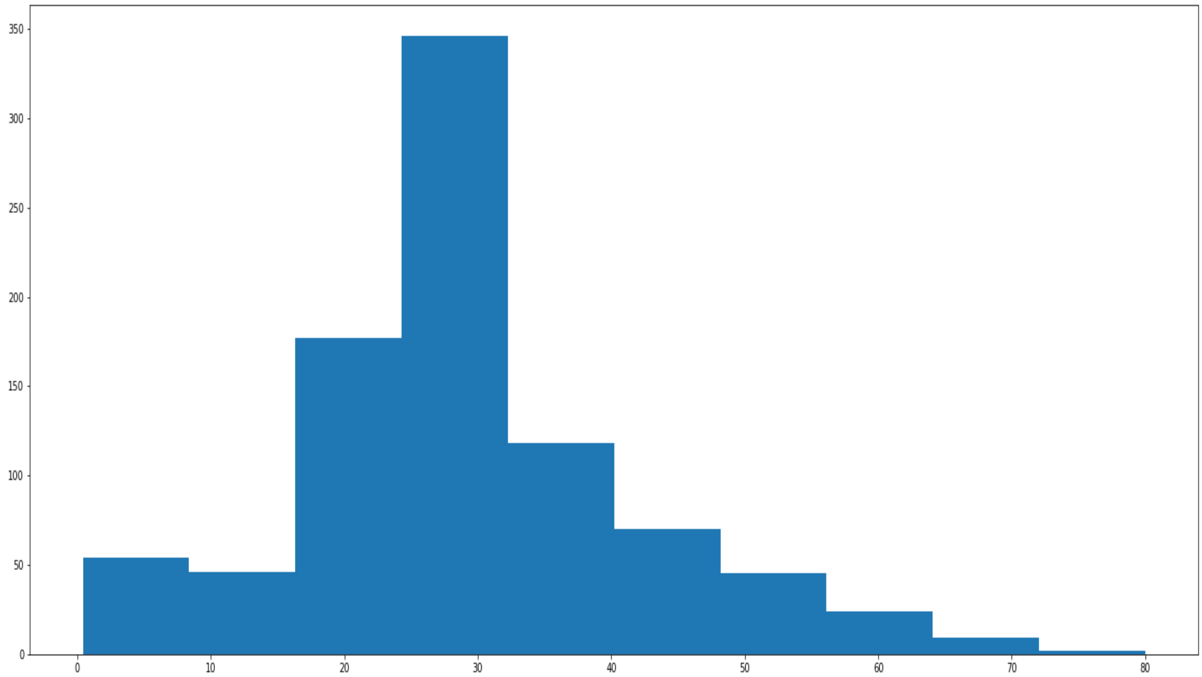
This result shows the number of missing values at each column. I dropped the columns 'Ticket' and 'Cabin'. I filled the mean value for all the missing values at 'Age' column. Mean value is 29.699. In case of 'Embarked' column, I just filled the missing value with 'S' because 'S' is the most in column.

```
df = df.fillna({'Age' : mean})
[redacted]

df.show()
df.printSchema()
```

```
+-----------+--------+------+--------------------+------+-----------------+-----+-----+-------+--------+
|PassengerId|Survived|Pclass|                Name|   Sex|              Age|SibSp|Parch|   Fare|Embarked|
+-----------+--------+------+--------------------+------+-----------------+-----+-----+-------+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|               22|    1|    0|   7.25|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female|               38|    1|    0|71.2833|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female|               26|    0|    0|  7.925|       S|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|               35|    1|    0|   53.1|       S|
|          5|       0|     3|Allen, Mr. Willia...|  male|               35|    0|    0|   8.05|       S|
|          6|       0|     3|    Moran, Mr. James|  male|29.69911764705882|    0|    0| 8.4583|       Q|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|               54|    0|    0|51.8625|       S|
|          8|       0|     3|Palsson, Master. ...|  male|                2|    3|    1| 21.075|       S|
|          9|       1|     3|Johnson, Mrs. Osc...|female|               27|    0|    2|11.1333|       S|
|         10|       1|     2|Nasser, Mrs. Nich...|female|               14|    1|    0|30.0708|       C|
|         11|       1|     3|Sandstrom, Miss. ...|female|                4|    1|    1|   16.7|       S|
|         12|       1|     1|Bonnell, Miss. El...|female|               58|    0|    0|  26.55|       S|
|         13|       0|     3|Saundercock, Mr. ...|  male|               20|    0|    0|   8.05|       S|
|         14|       0|     3|Andersson, Mr. An...|  male|               39|    1|    5| 31.275|       S|
|         15|       0|     3|Vestrom, Miss. Hu...|female|               14|    0|    0| 7.8542|       S|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|               55|    0|    0|     16|       S|
|         17|       0|     3|Rice, Master. Eugene|  male|                2|    4|    1| 29.125|       Q|
|         18|       1|     2|Williams, Mr. Cha...|  male|29.69911764705882|    0|    0|     13|       S|
|         19|       0|     3|Vander Planke, Mr...|female|               31|    1|    0|     18|       S|
|         20|       1|     3|Masselmani, Mrs. ...|female|29.69911764705882|    0|    0|  7.225|       C|
+-----------+--------+------+--------------------+------+-----------------+-----+-----+-------+--------+
only showing top 20 rows
```

1-(c)



Null values in ‘Age’ column are all changed to mean value 29.699, so we can see that there are many cases near value 30.

1-(d)

SexIndex	PclassIndex	AgeIndex	SibspIndex	ParchIndex	EmbarkedIndex
0.0	0.0	2.0	1.0	0.0	0.0
1.0	1.0	26.0	1.0	0.0	1.0
1.0	0.0	11.0	0.0	0.0	0.0
1.0	1.0	14.0	1.0	0.0	0.0
0.0	0.0	14.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	2.0
0.0	1.0	33.0	0.0	0.0	0.0
0.0	0.0	27.0	4.0	1.0	0.0
1.0	0.0	12.0	0.0	2.0	0.0
1.0	2.0	37.0	1.0	0.0	1.0
1.0	0.0	28.0	1.0	1.0	0.0
1.0	1.0	45.0	0.0	0.0	0.0
0.0	0.0	17.0	0.0	0.0	0.0
0.0	0.0	21.0	1.0	4.0	0.0
1.0	0.0	37.0	0.0	0.0	0.0
1.0	2.0	66.0	0.0	0.0	0.0
0.0	0.0	27.0	3.0	1.0	2.0
0.0	2.0	0.0	0.0	0.0	0.0
1.0	0.0	16.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	1.0

SexHot	PclassHot	AgeHot	SibspHot	ParchHot	EmbarkedHot
(1,[0],[1.0])	(2,[0],[1.0])	(88,[2],[1.0])	(6,[1],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[1],[1.0])	(88,[26],[1.0])	(6,[1],[1.0])	(6,[0],[1.0])	(3,[1],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[11],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[1],[1.0])	(88,[14],[1.0])	(6,[1],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[14],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[0],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[2],[1.0])
(1,[0],[1.0])	(2,[1],[1.0])	(88,[33],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[27],[1.0])	(6,[4],[1.0])	(6,[1],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[12],[1.0])	(6,[0],[1.0])	(6,[2],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[],[])	(88,[37],[1.0])	(6,[1],[1.0])	(6,[0],[1.0])	(3,[1],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[28],[1.0])	(6,[1],[1.0])	(6,[1],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[1],[1.0])	(88,[45],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[17],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[21],[1.0])	(6,[1],[1.0])	(6,[4],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[37],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[],[])	(88,[66],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[0],[1.0])	(2,[0],[1.0])	(88,[27],[1.0])	(6,[3],[1.0])	(6,[1],[1.0])	(3,[2],[1.0])
(1,[0],[1.0])	(2,[],[])	(88,[0],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[16],[1.0])	(6,[1],[1.0])	(6,[0],[1.0])	(3,[0],[1.0])
(1,[],[])	(2,[0],[1.0])	(88,[0],[1.0])	(6,[0],[1.0])	(6,[0],[1.0])	(3,[1],[1.0])

I used StringIndexer and OneHotEncoder at columns ['Sex', 'PClass', 'Age', 'Sibsp', 'Parch', 'Embarked'].

1-(e)

```
[39] train, test = df_transformed.randomSplit([0.8, 0.2], seed=None)

     train = train.withColumn('testOrtrain', lit('train'))
     test = test.withColumn('testOrtrain', lit('test'))

     train.show()
     test.show()
```

2-(a)

```
pipeline = Pipeline(stages=[sex_indexer,
                            pclass_indexer,
                            age_indexer,
                            sibsp_indexer,
                            parch_indexer,
                            embarked_indexer,
                            onehotencoder_sex_vector,
                            onehotencoder_pclass_vector,
                            onehotencoder_age_vector,
                            onehotencoder_sibsp_vector,
                            onehotencoder_parch_vector,
                            onehotencoder_embarked_vector
                    ])
```

```
from pyspark.ml.feature import VectorAssembler

va = VectorAssembler(inputCols=['SexHot',
                                'PclassHot',
                                'AgeHot',
                                'SibspHot',
                                'ParchHot',
                                'EmbarkedHot'],
                     outputCol='vector')

pipeline = Pipeline(stages=[va])
```

```
lr = (LogisticRegression().
      setLabelCol('Survived').
      setFeaturesCol('vector').
      setRegParam(0.0).
      setMaxIter(100).
      setElasticNetParam(0.)
      )
```

2-(b, c)

```
[37] from pyspark.ml.classification import LogisticRegression
     from pyspark.ml.evaluation import BinaryClassificationEvaluator
     from pyspark.sql import functions as F

     lr = (LogisticRegression().
           setLabelCol('Survived').
           setFeaturesCol('vector').
           setRegParam(0.0).
           setMaxIter(100).
           setElasticNetParam(0.)
           )

     lrmodel = lr.fit(train)

     lrDf = lrmodel.transform(test)
     lrDf.groupBy('prediction', 'Survived').count().show()

     evaluator = BinaryClassificationEvaluator(rawPredictionCol = 'prediction', labelCol='Survived')
     print(evaluator.evaluate(lrDf)*100 , '%')
```

```
+----------+--------+-----+
|prediction|Survived|count|
+----------+--------+-----+
|       1.0|       0|   13|
|       0.0|       0|   89|
|       0.0|       1|   21|
|       1.0|       1|   41|
+----------+--------+-----+

76.69196710942441 %
```

```
from sklearn.metrics import precision_score
survived_list = list(lrDf.select('Survived').toPandas()['Survived'])
prediction_list = list(lrDf.select('prediction').toPandas()['prediction'])

print("precision ", precision_score(survived_list, prediction_list, average='micro'))
```

```
precision  0.7926829268292683
```

First I fit the model and evaluate the model with BinaryClassificationEvaluator. Accuracy is about 76.7%. When we use precision score as our evaluation metric, precision is about 0.79. It's not bad score.

2-(d)

There are totally 105 coefficients corresponding to each feature of data.

```
coefficients:  [-3.0029383765608,-0.963620018107391,1.2198368074488808,-
19.320025512558495,-19.527120410663006,-19.424382496376726,-
20.15676925355426,-18.703935425747712,-19.88024192746456,-
```

```
19.98909694282752,-20.257760618680337,-19.841952707734333,-
18.8209151445931,-18.84849879916882,-18.957092740393456,-
18.34344923406654,-16.967662998292383,-18.505375072643307,-
18.626100779575236,-19.166003330281903,-18.73253400876625,-
19.15551264963913,-19.844075854796216,-19.039358108308058,-
19.484242684368116,-19.534090183450044,-19.57455982849251,-
18.99094754687771,-19.576464922828162,-19.77692576268075,-
19.841973562723048,-14.95541535060405,-19.255920444302067,-
17.882817524605393,-113.53177458958828,-18.68141400147053,-
19.93404187269865,-18.600098223627306,114.45835284631684,-
19.551389781778223,-19.721362898317174,-13.746540102446753,-
20.622390877655764,-20.69022609870715,-18.754149094956222,-
19.55629969521454,-18.118820131491645,-98.24989615309896,-
20.445260291679055,-44.4999861056707,194.26653519028946,-
19.35842144473586,-19.66033435391316,-19.210514773089166,-
16.59657029109311,-113.25331173787293,-16.624782246675665,-
115.17863566565113,-113.57099190239435,-
17.638197188662225,102.51363448809668,0.0,-
114.97151548445754,75.54817403488714,-83.12822054803078,-
138.95811942674047,-19.37471548743924,-100.44082352162687,-
116.59872542584401,-19.84827699760172,-122.46178829001612,-
98.89596521286074,75.20870412603597,-13.0928044924888,-
109.57790916667467,-
113.52354844249308,111.38745204917868,109.93301903808364,106.8961651007
539,0.0,-105.8436178590594,-83.12822054803078,-89.29538305810833,-
83.12822054803078,-89.29538305810833,-
107.53082634849298,59.717476386874424,-83.12822054803078,-
106.63471211218928,-114.2845265179959,-
83.12822054803078,11.957293617921755,11.828385760041716,11.549021120944
003,8.089751072765697,6.492830899374069,-
88.69606132658289,7.5742519867940965,8.015398027988477,7.70076133454758
25,7.359051885263463,7.137739684856806,-104.92742853273096,-
0.6121165724456985,-0.21025339501315088]
```

```
intercept:  1.8203373972189798
```

And the higher value of coefficient, the feature has big impact to model's prediction. Of course, if coefficient is positive, It affects prediction of target feature to 1. While if coefficient is negative, It affects prediction of target feature to 0. Therefore, we can say that the feature which has largest absolute value of coefficients contributes the learning most. 88 coefficients of total 105 coefficients are related to 'Age' attribute. I seek the average of coefficients which belong to each attribute. Average of coefficients is largest at 'Age' attribute. Although the result is different at each randomsplit and execution, avg of 'Age' coefficients is largest most frequently. Top 3 largest attributes are as follows.

```
avg of 'Age' coefficients
50.829271457413846
```

```
avg of 'Sibsp' coefficients
23.10222396627169
```

```
avg of 'Parch' coefficients
23.92283356994004
```

`Code)` # I executed code in google Colab. I also attached my .ipynb file. You should upload train.csv file to Colab.

```
!pip install pyspark

from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark import SparkContext
import pandas as pd

spark = SparkSession.builder.appName("Titanic-Dataset").config('spark.driver.memory','15g').getOrCreate()

df = spark.read.option("header", True).csv('train.csv')
df.printSchema()

from pyspark.sql.functions import when, count, col
df.select([count(when(isnull(c), c)).alias(c) for c in df.columns]).show()

columns_to_drop = ['Ticket', 'Cabin']
df = df.drop(*columns_to_drop)

from pyspark.sql.functions import mean as _mean, stddev as _stddev, col
```

```
df_stats = df.select(
    _mean(col('Age')).alias('mean'),
).collect()

mean = df_stats[0]['mean']
print(mean)

df = df.fillna({'Age' : mean})

df.show()
df.printSchema()

df = df.fillna({'Embarked': '0'})
df_pd = df.toPandas()
age_list = list(df_pd['Age'])

age_list = [float (i) for i in age_list]
age_list.sort()

import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (25, 10)
plt.hist(age_list)
plt.show()

from pyspark.ml.feature import StringIndexer
```

```
from pyspark.ml.feature import OneHotEncoder
from pyspark.ml import Pipeline

sex_indexer = StringIndexer(inputCol="Sex", outputCol="SexIndex")
pclass_indexer = StringIndexer(inputCol="Pclass", outputCol="PclassIndex")
age_indexer = StringIndexer(inputCol="Age", outputCol="AgeIndex")
sibsp_indexer = StringIndexer(inputCol="SibSp", outputCol="SibspIndex")
parch_indexer = StringIndexer(inputCol="Parch", outputCol="ParchIndex")
embarked_indexer = StringIndexer(inputCol="Embarked", outputCol="EmbarkedIndex")
onehotencoder_sex_vector = OneHotEncoder(inputCol="SexIndex", outputCol="SexHot")
onehotencoder_pclass_vector = OneHotEncoder(inputCol="PclassIndex", outputCol="PclassHot")
onehotencoder_age_vector = OneHotEncoder(inputCol="AgeIndex", outputCol="AgeHot")
onehotencoder_sibsp_vector = OneHotEncoder(inputCol="SibspIndex", outputCol="SibspHot")
onehotencoder_parch_vector = OneHotEncoder(inputCol="ParchIndex", outputCol="ParchHot")
onehotencoder_embarked_vector = OneHotEncoder(inputCol="EmbarkedIndex",
outputCol="EmbarkedHot")

pipeline = Pipeline(stages=[sex_indexer,
                            pclass_indexer,
                            age_indexer,
                            sibsp_indexer,
                            parch_indexer,
                            embarked_indexer,
                            onehotencoder_sex_vector,
                            onehotencoder_pclass_vector,
                            onehotencoder_age_vector,
```

```
                    onehotencoder_sibsp_vector,
                    onehotencoder_parch_vector,
                    onehotencoder_embarked_vector
                ])

df_transformed = pipeline.fit(df).transform(df)
df_transformed.show()

from pyspark.sql.types import IntegerType

df_transformed = df_transformed.withColumn("Survived", df_transformed["Survived"].cast(IntegerType()))


from pyspark.ml.feature import VectorAssembler

va = VectorAssembler(inputCols=['SexHot',
                                'PclassHot',
                                'AgeHot',
                                'SibspHot',
                                'ParchHot',
                                'EmbarkedHot'],
                     outputCol='vector')

pipeline = Pipeline(stages=[va])
```

```
df_transformed = pipeline.fit(df_transformed).transform(df_transformed)
df_transformed.show()

train, test = df_transformed.randomSplit([0.8, 0.2], seed=None)

train = train.withColumn('testOrtrain', lit('train'))
test = test.withColumn('testOrtrain', lit('test'))

train.show()
test.show()

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql import functions as F

lr = (LogisticRegression().
      setLabelCol('Survived').
      setFeaturesCol('vector').
      setRegParam(0.0).
      setMaxIter(100).
      setElasticNetParam(0.)
      )

lrmodel = lr.fit(train)

lrDf = lrmodel.transform(test)
```

```
lrDf.groupBy('prediction', 'Survived').count().show()

evaluator = BinaryClassificationEvaluator(rawPredictionCol = 'prediction', labelCol='Survived')
print(evaluator.evaluate(lrDf)*100 , '%')

from sklearn.metrics import precision_score
survived_list = list(lrDf.select('Survived').toPandas()['Survived'])
prediction_list = list(lrDf.select('prediction').toPandas()['prediction'])

print("precision ", precision_score(survived_list, prediction_list, average='micro'))

print("coefficients:", lrmodel.coefficients)

print("intercept: ", lrmodel.intercept)
print()

print("avg of 'Age' coefficients")
age_list = [float (i) for i in lrmodel.coefficients[3:91]]
sum = 0
for a in age_list:
  if a<0:
    sum+=(a*-1)
  else:
    sum+=a
print(sum/88)
```

```
print("avg of 'Sibsp' coefficients")
age_list = [float (i) for i in lrmodel.coefficients[91:97]]
sum = 0
for a in age_list:
   if a<0:
      sum+=(a*-1)
   else:
      sum+=a
print(sum/6)

print("avg of 'Parch' coefficients")
age_list = [float (i) for i in lrmodel.coefficients[97:103]]
sum = 0
for a in age_list:
   if a<0:
      sum+=(a*-1)
   else:
      sum+=a
print(sum/6)
```