databricks | Search CTRL + P | CMPE 255 | kisung.park@sjsu.edu

Get started

This is your home for all data science and engineering work.

We'll show you how to set up clusters, data and users.

Set up your workspace

- Create a cluster
- Import data
- Build a Delta Live Tables pipeline
- Invite your team

Next steps

- Explore data with Python
- Read documentation

2/4

Don't show again

titanic

Python

File Edit View Run Help Last edit was 7 minutes ago Give feedback

Run all | [default]basic-starter-cl... | Schedule | Share

Free trial ends in 14 days. Continue with a pay-as-you-go subscription by providing your billing information.

Cmd 1

```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style

# Algorithms
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
```

Command took 0.06 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 2

```
test_df = pd.read_csv("/dbfs/FileStore/tables/test.csv")
train_df = pd.read_csv("/dbfs/FileStore/tables/train.csv")
```

Command took 0.19 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 3

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 4

```
train_df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 5

```
train_df.head(15)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 6

```
total = train_df.isnull().sum().sort_values(ascending=False)
percent_1 = train_df.isnull().sum()/train_df.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(5)
```

	Total	%
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2
PassengerId	0	0.0
Survived	0	0.0

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 7

```
train_df.columns.values
```

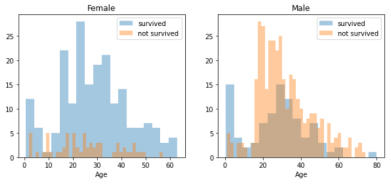
```
Out[34]: array(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'], dtype=object)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 8

```
survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 4))
women = train_df[train_df['Sex']=='female']
men = train_df[train_df['Sex']=='male']
ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[0], kde =False)
ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[0], kde =False)
ax.legend()
ax.set_title('Female')
ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False)
ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[1], kde = False)
ax.legend()
_ = ax.set_title('Male')
```

```
/databricks/python/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

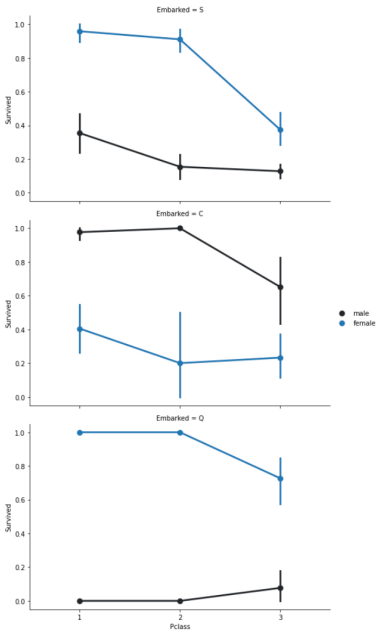


Command took 0.90 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 9

```
1  FacetGrid = sns.FacetGrid(train_df, row='Embarked', size=4.5, aspect=1.6)
2  FacetGrid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette=None,  order=None, hue_order=None )
3  FacetGrid.add_legend()
```

```
/databricks/python/lib/python3.9/site-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
Out[36]: <seaborn.axisgrid.FacetGrid at 0x7f38e9ec6220>
```

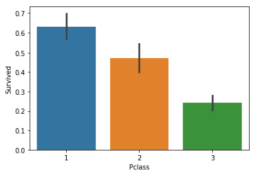


Command took 1.80 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 10

```
1  sns.barplot(x='Pclass', y='Survived', data=train_df)
```

```
Out[37]: <AxesSubplot:xlabel='Pclass', ylabel='Survived'>
```

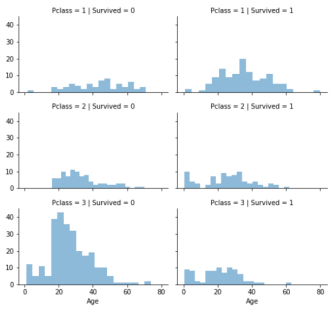


Command took 0.40 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 11

```
1  grid = sns.FacetGrid(train_df, col='Survived', row='Pclass', size=2.2, aspect=1.6)
2  grid.map(plt.hist, 'Age', alpha=.5, bins=20)
3  grid.add_legend();
```

```
/databricks/python/lib/python3.9/site-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



Command took 1.80 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 12

```
1  data = [train_df, test_df]
2  for dataset in data:
3      dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
4      dataset.loc[dataset['relatives'] > 0, 'not_alone'] = 0
5      dataset.loc[dataset['relatives'] == 0, 'not_alone'] = 1
6      dataset['not_alone'] = dataset['not_alone'].astype(int)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 13

```
1  train_df['not_alone'].value_counts()
```

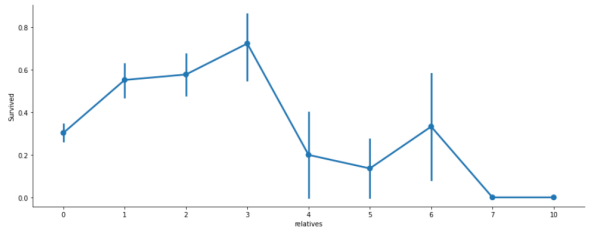
```
Out[40]: 1    537
0    354
Name: not_alone, dtype: int64
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 14

```
axes = sns.factorplot('relatives','Survived',
                      data=train_df, aspect = 2.5, )
```

```
/databricks/python/lib/python3.9/site-packages/seaborn/categorical.py:3717: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
/databricks/python/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Command took 0.79 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 15

```
train_df = train_df.drop(['PassengerId'], axis=1)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 16

```
import re
deck = {"A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6, "G": 7, "U": 8}
data = [train_df, test_df]

for dataset in data:
    dataset['Cabin'] = dataset['Cabin'].fillna("U0")
    dataset['Deck'] = dataset['Cabin'].map(lambda x: re.compile("([a-zA-Z]+)").search(x).group())
    dataset['Deck'] = dataset['Deck'].map(deck)
    dataset['Deck'] = dataset['Deck'].fillna(0)
    dataset['Deck'] = dataset['Deck'].astype(int)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 17

```
# we can now drop the cabin feature
train_df = train_df.drop(['Cabin'], axis=1)
test_df = test_df.drop(['Cabin'], axis=1)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 18

```
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)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 19

```
train_df["Age"].isnull().sum()
```

```
Out[46]: 0
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 20

```
train_df['Embarked'].describe()
```

```
Out[47]: count     889
unique      3
top         S
freq      644
Name: Embarked, dtype: object
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 21

```
common_value = 'S'
data = [train_df, test_df]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna(common_value)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 22

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    891 non-null    int64
 1   Pclass      891 non-null    int64
 2   Name        891 non-null    object
 3   Sex         891 non-null    object
 4   Age         891 non-null    int64
 5   SibSp       891 non-null    int64
 6   Parch       891 non-null    int64
 7   Ticket      891 non-null    object
 8   Fare        891 non-null    float64
 9   Embarked    891 non-null    object
 10  relatives   891 non-null    int64
 11  not_alone   891 non-null    int64
 12  Deck        891 non-null    int64
dtypes: float64(1), int64(8), object(4)
memory usage: 90.6+ KB
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 23

```
data = [train_df, test_df]

for dataset in data:
    dataset['Fare'] = dataset['Fare'].fillna(0)
    dataset['Fare'] = dataset['Fare'].astype(int)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 24

```
data = [train_df, test_df]
titles = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

for dataset in data:
    # extract titles
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
    # replace titles with a more common title or as Rare
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr',\
                                            'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
    # convert titles into numbers
    dataset['Title'] = dataset['Title'].map(titles)
```

```
    # filling NaN with 0, to get safe
    dataset['Title'] = dataset['Title'].fillna(0)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 25

```
train_df = train_df.drop(['Name'], axis=1)
test_df = test_df.drop(['Name'], axis=1)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 26

```
genders = {"male": 0, "female": 1}
data = [train_df, test_df]

for dataset in data:
    dataset['Sex'] = dataset['Sex'].map(genders)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 27

```
train_df['Ticket'].describe()
```

```
Out[54]: count        891
unique        681
top        347082
freq            7
Name: Ticket, dtype: object
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 28

```
train_df = train_df.drop(['Ticket'], axis=1)
test_df = test_df.drop(['Ticket'], axis=1)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 29

```
ports = {"S": 0, "C": 1, "Q": 2}
data = [train_df, test_df]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].map(ports)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 30

```
data = [train_df, test_df]
for dataset in data:
    dataset['Age'] = dataset['Age'].astype(int)
    dataset.loc[ dataset['Age'] <= 11, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 11) & (dataset['Age'] <= 18), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 18) & (dataset['Age'] <= 22), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 22) & (dataset['Age'] <= 27), 'Age'] = 3
    dataset.loc[(dataset['Age'] > 27) & (dataset['Age'] <= 33), 'Age'] = 4
    dataset.loc[(dataset['Age'] > 33) & (dataset['Age'] <= 40), 'Age'] = 5
    dataset.loc[(dataset['Age'] > 40) & (dataset['Age'] <= 66), 'Age'] = 6
    dataset.loc[ dataset['Age'] > 66, 'Age'] = 6
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 31

```
# let's see how it's distributed
train_df['Age'].value_counts()
```

```
Out[58]: 6    165
4    153
5    147
3    143
2    117
1     98
0     68
Name: Age, dtype: int64
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 32

```
train_df.head(10)
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone	Deck	Title
0	0	3	0	2	1	0	7	0	1	0	8	1
1	1	1	1	5	1	0	71	1	1	0	3	3
2	1	3	1	3	0	0	7	0	0	1	8	2
3	1	1	1	5	1	0	53	0	1	0	3	3
4	0	3	0	5	0	0	8	0	0	1	8	1
5	0	3	0	3	0	0	8	2	0	1	8	1
6	0	1	0	6	0	0	51	0	0	1	5	1
7	0	3	0	0	3	1	21	0	4	0	8	4
8	1	3	1	3	0	2	11	0	2	0	8	3
9	1	2	1	1	1	0	30	1	1	0	8	3

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 33

```
data = [train_df, test_df]

for dataset in data:
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare'] = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   = 2
    dataset.loc[(dataset['Fare'] > 31) & (dataset['Fare'] <= 99), 'Fare']   = 3
    dataset.loc[(dataset['Fare'] > 99) & (dataset['Fare'] <= 250), 'Fare']   = 4
    dataset.loc[ dataset['Fare'] > 250, 'Fare'] = 5
    dataset['Fare'] = dataset['Fare'].astype(int)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 34

```
data = [train_df, test_df]
for dataset in data:
    dataset['Age_Class']= dataset['Age']* dataset['Pclass']
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 35

```
for dataset in data:
    dataset['Fare_Per_Person'] = dataset['Fare']/(dataset['relatives']+1)
    dataset['Fare_Per_Person'] = dataset['Fare_Per_Person'].astype(int)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 36

```
# Let's take a last look at the training set, before we start training the models.
train_df.head(20)
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	relatives	not_alone	Deck	Title	Age_Class	Fare_Per_Person
0	0	3	0	2	1	0	0	0	1	0	8	1	6	0
1	1	1	1	5	1	0	3	1	1	0	3	3	5	1
2	1	3	1	3	0	0	0	0	0	1	8	2	9	0
3	1	1	1	5	1	0	3	0	1	0	3	3	5	1
4	0	3	0	5	0	0	1	0	0	1	8	1	15	1
5	0	3	0	3	0	0	1	2	0	1	8	1	9	1
6	0	1	0	6	0	0	3	0	0	1	5	1	6	3
7	0	3	0	0	3	1	2	0	4	0	8	4	0	0
8	1	3	1	3	0	2	1	0	2	0	8	3	9	0
9	1	2	1	1	1	0	2	1	1	0	8	3	2	1
10	1	3	1	0	1	1	2	0	2	0	7	2	0	0
11	1	1	1	6	0	0	2	0	0	1	3	2	6	2
12	0	3	0	2	0	0	1	0	0	1	8	1	6	1
13	0	3	0	5	1	5	2	0	6	0	8	1	15	0
14	0	3	1	1	0	0	0	0	0	1	8	2	3	0

15	1	2	1	6	0	0	2	0	0	1	8	3	12	2
16	0	3	0	0	4	1	2	2	5	0	8	4	0	0
17	1	2	0	5	0	0	1	0	0	1	8	1	10	1
18	0	3	1	4	1	0	2	0	1	0	8	3	12	1
19	1	3	1	5	0	0	0	1	0	1	8	3	15	0

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 37

```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test  = test_df.drop("PassengerId", axis=1).copy()
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 38

```
# stochastic gradient descent (SGD) learning
sgd = linear_model.SGDClassifier(max_iter=5, tol=None)
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)

sgd.score(X_train, Y_train)

acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)


print(round(acc_sgd,2,), "%")
```

```
77.89 %
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 39

```
# Random Forest
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)

Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
print(round(acc_random_forest,2,), "%")
```

```
92.48 %
```

Command took 0.39 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 40

```
# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)

Y_pred = logreg.predict(X_test)

acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
print(round(acc_log,2,), "%")
```

```
81.59 %
/databricks/python/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Command took 0.20 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 41

```
# KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)

Y_pred = knn.predict(X_test)

acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
print(round(acc_knn,2,), "%")
```

```
86.31 %
```

Command took 0.19 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 42

```
# Gaussian Naive Bayes
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)

Y_pred = gaussian.predict(X_test)

acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
print(round(acc_gaussian,2,), "%")
```

```
77.44 %
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 43

```
# Perceptron
perceptron = Perceptron(max_iter=5)
perceptron.fit(X_train, Y_train)

Y_pred = perceptron.predict(X_test)

acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
print(round(acc_perceptron,2,), "%")
```

```
63.75 %
/databricks/python/lib/python3.9/site-packages/sklearn/linear_model/_stochastic_gradient.py:574: ConvergenceWarning: Maximum number of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 44

```
# Linear SVC
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)

Y_pred = linear_svc.predict(X_test)

acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
print(round(acc_linear_svc,2,), "%")
```

```
81.48 %
/databricks/python/lib/python3.9/site-packages/sklearn/svm/_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

Command took 0.19 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 45

```
# Decision Tree
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)

Y_pred = decision_tree.predict(X_test)

acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
print(round(acc_decision_tree,2,), "%")
```

```
92.48 %
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 46

```
results = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent',
              'Decision Tree'],
    'Score': [acc_linear_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_decision_tree]})
result_df = results.sort_values(by='Score', ascending=False)
result_df = result_df.set_index('Score')
result_df.head(9)
```

Score	Model
92.48	Random Forest
92.48	Decision Tree
86.31	KNN
81.59	Logistic Regression
81.48	Support Vector Machines
77.89	Stochastic Gradient Decent
77.44	Naive Bayes
63.75	Perceptron

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 47

```
from sklearn.model_selection import cross_val_score
rf = RandomForestClassifier(n_estimators=100)
scores = cross_val_score(rf, X_train, Y_train, cv=10, scoring = "accuracy")
```

Command took 2.20 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 48

```
print("Scores:", scores)
print("Mean:", scores.mean())
print("Standard Deviation:", scores.std())
```

```
Scores: [0.77777778 0.82022472 0.74157303 0.84269663 0.85393258 0.85393258
 0.79775281 0.7752809  0.83146067 0.83146067]
Mean: 0.8126092384519351
Standard Deviation: 0.03600379031824795
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 49

```
importances = pd.DataFrame({'feature':X_train.columns,'importance':np.round(random_forest.feature_importances_,3)})
importances = importances.sort_values('importance',ascending=False).set_index('feature')
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 50

```
importances.head(15)
```

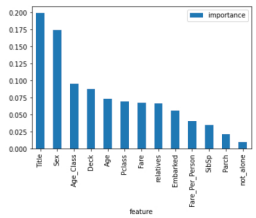
feature	importance
Title	0.199
Sex	0.174
Age_Class	0.096
Deck	0.088
Age	0.073
Pclass	0.070
Fare	0.068
relatives	0.067
Embarked	0.056
Fare_Per_Person	0.041
SibSp	0.035
Parch	0.022
not_alone	0.010

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 51

```
importances.plot.bar()
```

```
Out[78]: <AxesSubplot:xlabel='feature'>
```



Command took 0.40 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 52

```
train_df  = train_df.drop("not_alone", axis=1)
test_df  = test_df.drop("not_alone", axis=1)

train_df  = train_df.drop("Parch", axis=1)
test_df  = test_df.drop("Parch", axis=1)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 53

```
# Random Forest

random_forest = RandomForestClassifier(n_estimators=100, oob_score = True)
random_forest.fit(X_train, Y_train)
Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)

acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
print(round(acc_random_forest,2,), "%")
```

```
92.48 %
```

Command took 0.60 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 54

```
print("oob score:", round(random_forest.oob_score_, 4)*100, "%")
```

```
oob score: 81.82000000000001 %
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 55

```
# Random Forest
random_forest = RandomForestClassifier(criterion = "gini",
                                       min_samples_leaf = 1,
                                       min_samples_split = 10,
                                       n_estimators=100,
                                       max_features='auto',
                                       oob_score=True,
                                       random_state=1,
                                       n_jobs=-1)

random_forest.fit(X_train, Y_train)
Y_prediction = random_forest.predict(X_test)

random_forest.score(X_train, Y_train)

print("oob score:", round(random_forest.oob_score_, 4)*100, "%")
```

```
oob score: 82.49 %
```

Command took 0.49 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 56

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
predictions = cross_val_predict(random_forest, X_train, Y_train, cv=3)
confusion_matrix(Y_train, predictions)
```

```
Out[83]: array([[487,  62],
       [101, 241]])
```

Command took 2.30 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 57

```
from sklearn.metrics import precision_score, recall_score

print("Precision:", precision_score(Y_train, predictions))
print("Recall:",recall_score(Y_train, predictions))
```

```
Precision: 0.7953795379537953
Recall: 0.7046783625730995
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 58

```
from sklearn.metrics import f1_score
f1_score(Y_train, predictions)
```

```
Out[85]: 0.7472868217054265
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

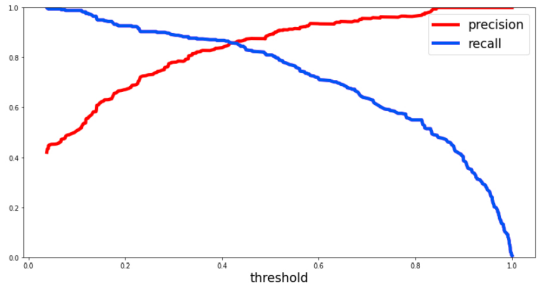
Cmd 59

```
from sklearn.metrics import precision_recall_curve

# getting the probabilities of our predictions
y_scores = random_forest.predict_proba(X_train)
y_scores = y_scores[:,1]

precision, recall, threshold = precision_recall_curve(Y_train, y_scores)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 60

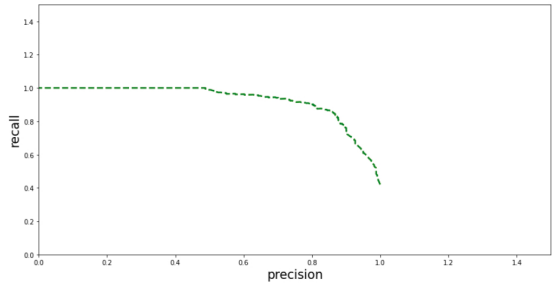
```
def plot_precision_and_recall(precision, recall, threshold):
    plt.plot(threshold, precision[:-1], "r-", label="precision", linewidth=5)
    plt.plot(threshold, recall[:-1], "b", label="recall", linewidth=5)
    plt.xlabel("threshold", fontsize=19)
    plt.legend(loc="upper right", fontsize=19)
    plt.ylim([0, 1])

plt.figure(figsize=(14, 7))
plot_precision_and_recall(precision, recall, threshold)
plt.show()
```



Command took 0.30 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 61

```
def plot_precision_vs_recall(precision, recall):
    plt.plot(recall, precision, "g--", linewidth=2.5)
    plt.ylabel("recall", fontsize=19)
    plt.xlabel("precision", fontsize=19)
    plt.axis([0, 1.5, 0, 1.5])

plt.figure(figsize=(14, 7))
plot_precision_vs_recall(precision, recall)
plt.show()
```



Command took 0.30 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

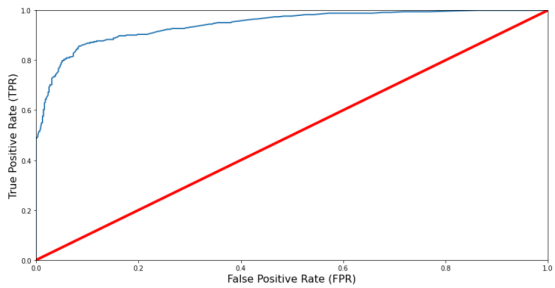
Cmd 62

```
from sklearn.metrics import roc_curve
# compute true positive rate and false positive rate
false_positive_rate, true_positive_rate, thresholds = roc_curve(Y_train, y_scores)
```

Command took 0.09 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 63

```
# plotting them against each other
def plot_roc_curve(false_positive_rate, true_positive_rate, label=None):
    plt.plot(false_positive_rate, true_positive_rate, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'r', linewidth=4)
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate (FPR)', fontsize=16)
    plt.ylabel('True Positive Rate (TPR)', fontsize=16)

plt.figure(figsize=(14, 7))
plot_roc_curve(false_positive_rate, true_positive_rate)
plt.show()
```



Command took 0.30 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 64

```
from sklearn.metrics import roc_auc_score
r_a_score = roc_auc_score(Y_train, y_scores)
print("ROC-AUC-Score:", r_a_score)
```

```
ROC-AUC-Score: 0.943070335218739
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Cmd 65

Python

```
submission = pd.DataFrame({
        "PassengerId": test_df["PassengerId"],
        "Survived": Y_prediction
    })
submission.to_csv('submission.csv', index=False)
```

Command took 0.10 seconds -- by kisung.park@sjsu.edu at 10/30/2022, 10:32:14 AM on [default]basic-starter-cluster

Shift+Enter to run