# Titanic Kaggle: Supervised machine Learning,

**Skills**: data pre-processing, handling nominal values, handling missing values, classification

**Summary**: This exercise focuses on preprocessing data before sending them to a classifier. You will deal with missing values, and with conversion of nominal values to vector format

**Data origin**: <https://www.kaggle.com/shreyask92/titanic-machine-learning-from-disaster/data>

## Basic data reading and inspection using the Pandas library

Let us open the data set with excel. This is feasible because the data set is relatively small. Then inspect the column names, as well as watch out for missing values. (The rest of the preprocessing part is based on code from <https://www.kaggle.com/shreyask92/titanic-machine-learning-from-disaster/notebook> with some modifications.)

## Some basic Pandas

## for the field/column Embarked it presents with True if the row has a null value, False if it does not have

trainData = pd.read\_csv('train.csv')

#it count all the records

len(trainData)

trainData['Embarked'].isnull()

#it counts the null values for the embarked field

trainData['Embarked'].isnull().sum()

#it counts the null values for all fields

trainData.isnull().sum()

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

#count the unique values of the field ‘PassangerID’

len(trainData['PassengerId'].unique())

#fill-in the missing values with the mean of the ‘age’ field

trainData.Age.fillna(value= trainData.Age.mean(),inplace=True)

## Data Preprocessing

import pandas as pd

import matplotlib.pyplot as plt

# Reading the train and the test data.

trainData = pd.read\_csv('train.csv')

testData = pd.read\_csv('test.csv')

# Displaying a sample of the train data to get more detailed info

trainData.head()

# if you cannot see all the columns, try

pd.set\_option('display.max\_columns', None)

trainData.head()

# to see the column names provide

trainData.columns

trainData.describe()

trainData.dtypes

PassengerId int64

Survived int64

Pclass int64

Name object

Sex object

Age float64

SibSp int64

Parch int64

Ticket object

Fare float64

Cabin object

Embarked object

## Missing values

Are there missing values in any of the columns? **True** denotes that there are missing values

trainData.isna().any()

PassengerId False

Survived False

Pclass False

Name False

Sex False

Age True

SibSp False

Parch False

Ticket False

Fare False

Cabin True

Embarked True

Are there missing values in any of the columns? True denotes that there are missing values

pd.DataFrame(trainData.isnull().sum() \* 100 / len(trainData))

## Feature selection: remove the target (class column), remove features with unique values, remove features with a high percentage of values

# Names of the features extracted from the data

selFeatures = list(trainData.columns.values)

# Removing the target variable from the column values

targetCol = 'Survived'

selFeatures.remove(targetCol)

# Removing features with unique values

for i in selFeatures:

if trainData.shape[0] == len(pd.Series(trainData[i]).unique()) :

selFeatures.remove(i)

# Removing features with high percentage of missing values

selFeatures.remove('Cabin')

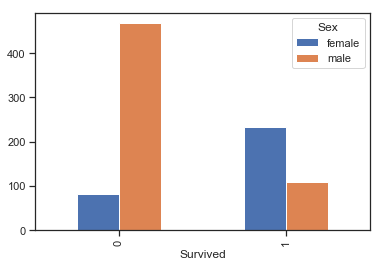
## Visualize data

import seaborn as sns

#plot the passengers that survived per sex

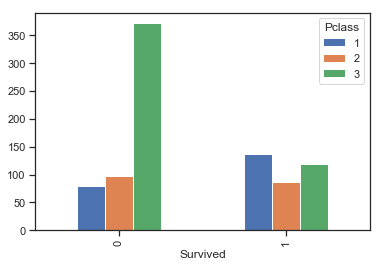
sex = pd.crosstab([trainData.Survived], trainData.Sex)

sex.plot.bar()



#TO-DO-1

# plot the number of passengersper class that survived



# Also removing cabin and ticket features for the initial run.

selFeatures.remove('Ticket')

print("Target Class: '"+ targetCol + "'")

print('Features to be investigated: ')

print(selFeatures)

## Handling missing values

Function to replace missing values in categorical data with a new value: X

def handle\_categorical\_na(df):

## replacing the null/na/nan values in 'Cabin' attribute with 'X'

df.Cabin = df.Cabin.fillna(value='X')

## Stripping the string data in 'Cabin' and 'Ticket' features of numeric values and duplicated characters

df.Cabin = [''.join(set(filter(str.isalpha, s))) for s in df.Cabin]

df.Ticket = [''.join(set(filter(str.isalpha, s))) for s in df.Ticket]

## replacing the '' values in 'Ticket' attribute with 'X'

df.Ticket.replace(to\_replace='',value='X',inplace=True)

## Imputing the null/na/nan values in 'Age' attribute with its mean value

df.Age.fillna(value=df.Age.mean(),inplace=True)

## replacing the null/na/nan values in 'Embarked' attribute with 'X'

df.Embarked.fillna(value='X',inplace=True)

return (df)

## Training and testing sets

from sklearn.model\_selection import train\_test\_split

seed = 7

np.random.seed(seed)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(trainData[selFeatures], trainData.Survived, test\_size=0.2)

X\_train = handle\_categorical\_na(X\_train)

X\_test = handle\_categorical\_na(X\_test)

## Use one Hot Encoding for categorical data

## using One Hot Encoding for handling categorical data

X\_train = pd.get\_dummies(X\_train,columns=['Embarked','Sex'],prefix=['Embarked','Sex'])

X\_test = pd.get\_dummies(X\_test,columns=['Embarked','Sex'],prefix=['Embarked','Sex'])