```
#LAB FINAL TASK NUMBER-1

import pandas as pd
train = pd.read_csv("/content/train.csv")
print("Dimensions of train: {}".format(train.shape))
```

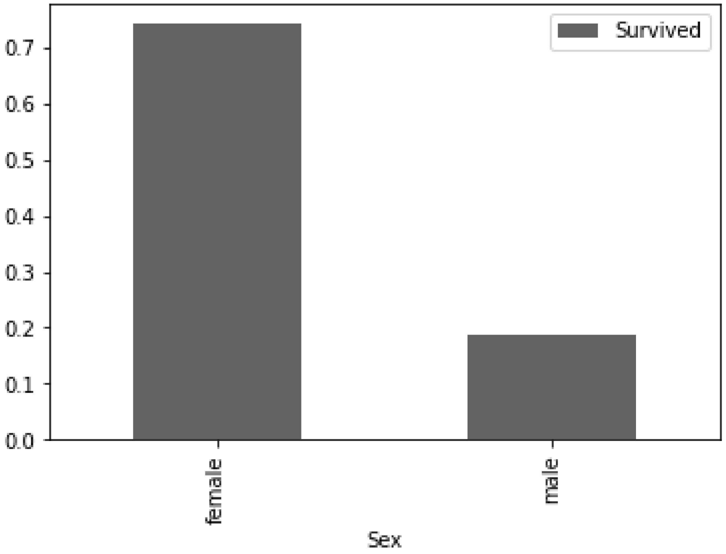
```
    Dimensions of train: (891, 12)
```

```
train.head(10)
```

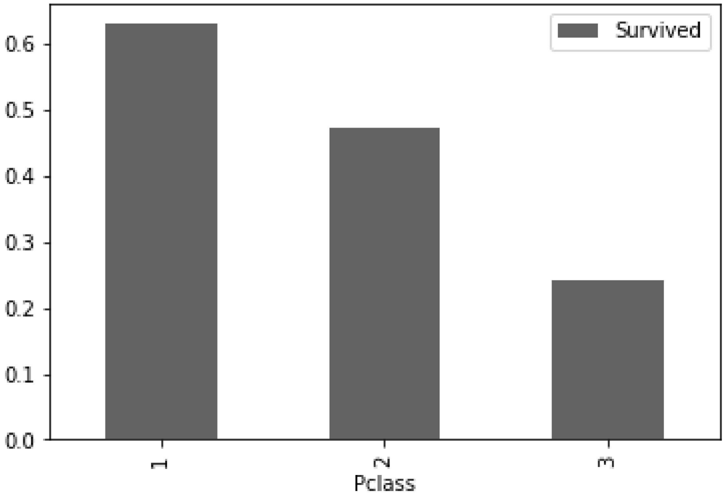
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

```
import matplotlib.pyplot as plt

sex_pivot = train.pivot_table(index="Sex",values="Survived")
sex_pivot.plot.bar()
plt.show()
```



```
class_pivot = train.pivot_table(index="Pclass",values="Survived")
class_pivot.plot.bar()
plt.show()
```

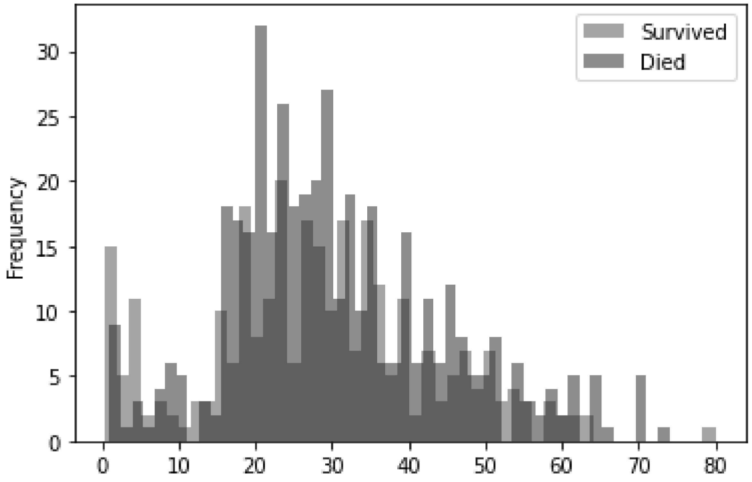


```
train["Age"].describe()
```

```
    count    714.000000
```

```
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

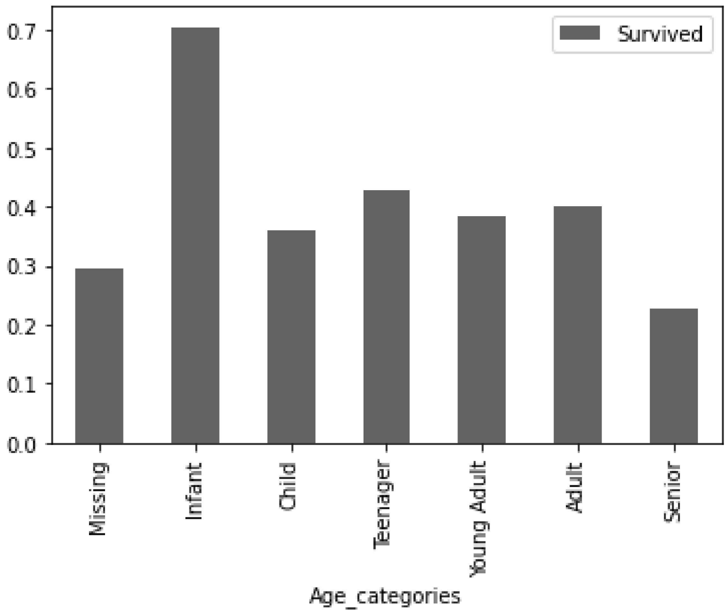
```
survived = train[train["Survived"] == 1]
died = train[train["Survived"] == 0]
survived["Age"].plot.hist(alpha=0.5,color='red',bins=50)
died["Age"].plot.hist(alpha=0.5,color='blue',bins=50)
plt.legend(['Survived','Died'])
plt.show()
```



```
def process_age(df,cut_points,label_names):
    df["Age"] = df["Age"].fillna(-0.5)
    df["Age_categories"] = pd.cut(df["Age"],cut_points,labels=label_names)
    return df

cut_points = [-1,0,5,12,18,35,60,100]
label_names = ["Missing","Infant","Child","Teenager","Young Adult","Adult","Senior"]
train = process_age(train,cut_points,label_names)
```

```
pivot = train.pivot_table(index="Age_categories",values='Survived')
pivot.plot.bar()
plt.show()
```



```
train["Pclass"].value_counts()
```

```
     3    491
     1    216
     2    184
     Name: Pclass, dtype: int64
```

```
def create_dummies(df,column_name):
    dummies = pd.get_dummies(df[column_name],prefix=column_name)
    df = pd.concat([df,dummies],axis=1)
    return df

for column in ["Pclass","Sex","Age_categories"]:
    train = create_dummies(train,column)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegression


lr = LogisticRegression()


columns = ['Pclass_2', 'Pclass_3', 'Sex_male']
lr.fit(train[columns], train['Survived'])
```

```
     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)
```

```
from sklearn.linear_model import LogisticRegression

columns = ['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male',
       'Age_categories_Missing','Age_categories_Infant',
       'Age_categories_Child', 'Age_categories_Teenager',
       'Age_categories_Young Adult', 'Age_categories_Adult',
      'Age_categories_Senior']

lr = LogisticRegression()
lr.fit(train[columns], train["Survived"])
```

```
     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                                  multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
                                  solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

```
from sklearn.model_selection import train_test_split

all_X = train[columns]
all_y = train['Survived']

train_X, test_X, train_y, test_y = train_test_split(
    all_X, all_y, test_size=0.20,random_state=0)
```

```
lr = LogisticRegression()
lr.fit(train_X, train_y)
predictions = lr.predict(test_X)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(test_y, predictions)
```

```
from sklearn.metrics import accuracy_score
lr = LogisticRegression()
lr.fit(train_X, train_y)
predictions = lr.predict(test_X)
accuracy = accuracy_score(test_y, predictions)
print(accuracy)
```

```
     0.8100558659217877
```

```
from sklearn.model_selection import cross_val_score

lr = LogisticRegression()
scores = cross_val_score(lr, all_X, all_y, cv=10)
scores.sort()
accuracy = scores.mean()

print(scores)
print(accuracy)
```

```
[0.76404494 0.76404494 0.76404494 0.78651685 0.8        0.80898876
 0.80898876 0.82022472 0.83146067 0.87640449]
0.8024719101123596
```

```
#LAB FINAL TASK NUMBER-2

def password_check(passwd):
    val = True

    if len(passwd) < 8:
        print('length should be at least 8')
        val = False

    if len(passwd) > 11:
        print('length should be not be greater than 10')
        val = False

    if not any(char.isdigit() for char in passwd[0]):
        print('Password should have start with a number')
        val = False

    if not any(char.isupper() for char in passwd[-2]):
        print('Password should have last 2 character must be capital')
        val = False
    if val:
        return val
def main():
    passwd = input("Enter your Password: ")

    if (password_check(passwd)):
        print("Valid")
    else:
        print("Invalid")

if __name__ == '__main__':
    main()
```

```
Enter your Password: 3asdaksAZ
```

```
    Valid
```

```
#LAB FINAL TASK NUMBER-3

totalModule = int(input("Enter the number of module: "))
devNumber = int(input("Enter the number of Developer: "))
days_per_one_module = int(input("Enter the number of days need a developer to complete 1 module: "))
cost_per_day = int(input("Enter the cost for one day: "))

print(f"{totalModule} number of module need to complete the Project.")

total_days = devNumber*days_per_one_module
print(f"{devNumber} devloper need to complete this task {total_days} days.")

total_cost = devNumber*cost_per_day*total_days
print(f"{devNumber} developer cost will be: {total_cost}")
```

```
    Enter the number of module: 25
    Enter the number of Developer: 5
    Enter the number of days need a developer to complete 1 module: 3
    Enter the cost for one day: 500
    25 number of module need to complete the Project.
    5 devloper need to complete this task 15 days.
    5 developer cost will be: 37500
```

✓ 0s completed at 10:49 PM ● ✕