

```
In [4]: # data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# machine learning
from sklearn.linear_model import LogisticRegression#, the dependent variable is a binary
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
In [6]: train_df = pd.read_csv("C:\\Users\\andre\\Desktop\\data\\train.csv")
test_df = pd.read_csv("C:\\Users\\andre\\Desktop\\data\\test.csv")
```

```
In [7]: combine = [train_df, test_df]
```

```
In [8]: print(train_df.columns.values)

['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

```
In [9]: train_df.info()
print('_'*40)
test_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
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  ---
418 non-null    int64
```

```
1 Pclass      418 non-null    int64
2 Name        418 non-null    object
3 Sex         418 non-null    object
4 Age         332 non-null    float64
5 SibSp       418 non-null    int64
6 Parch       418 non-null    int64
7 Ticket      418 non-null    object
8 Fare        417 non-null    float64
9 Cabin       91 non-null     object
10 Embarked    418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

```
In [10]: 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

```
In [11]: train_df.describe(include=['O'])
```

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Cohen, Mr. Gurshon "Gus"	male	1601	B96 B98	S
freq	1	577	7	4	644

```
In [12]: train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(b
```

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

```
In [13]: train_df[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().sort_values(by='Surv
```

	Sex	Survived
0	female	0.742038
1	male	0.188908

Out[14]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

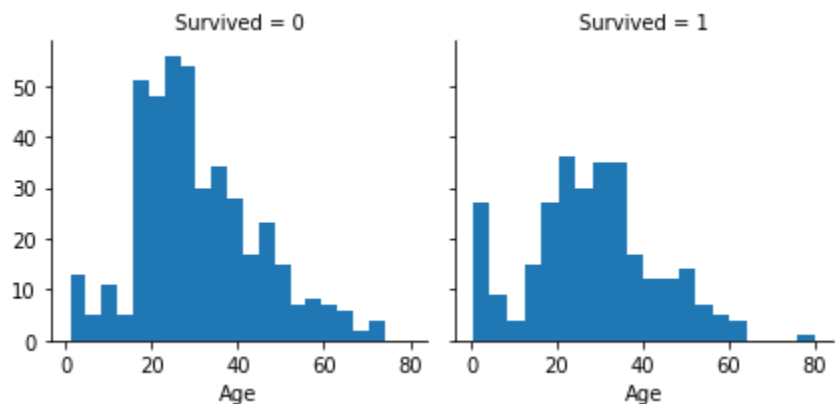
```
train_df[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mean().sort_values(by=
```

Out[15]:

	Parch	Survived
3	3	0.600000
1	1	0.550847
2	2	0.500000
0	0	0.343658
5	5	0.200000
4	4	0.000000
6	6	0.000000

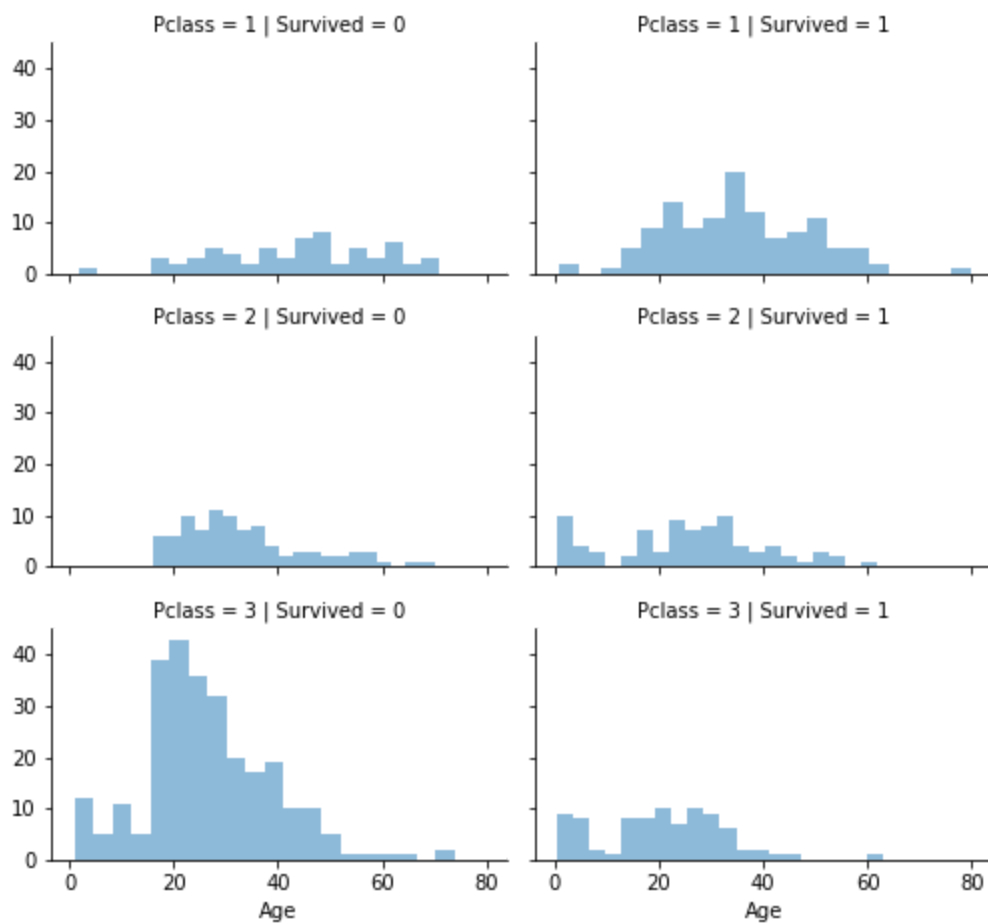
```
g = sns.FacetGrid(train_df, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x253ff845d00>



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

c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p  
y:316: UserWarning: The `size` parameter has been renamed to `height`; please update your  
code.  
warnings.warn(msg, UserWarning)

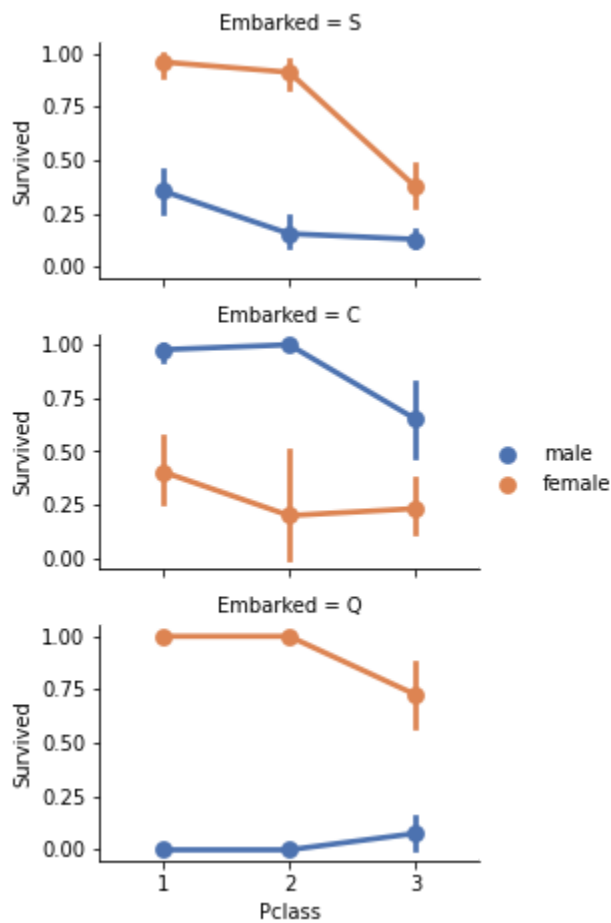


In [18]:

```
grid = sns.FacetGrid(train_df, row='Embarked', size=2.2, aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

```
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:316: UserWarning: The `size` parameter has been renamed to `height`; please update your
code.
  warnings.warn(msg, UserWarning)
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:643: UserWarning: Using the pointplot function without specifying `order` is likely to p
roduce an incorrect plot.
  warnings.warn(warning)
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:648: UserWarning: Using the pointplot function without specifying `hue_order` is likely
to produce an incorrect plot.
  warnings.warn(warning)
```

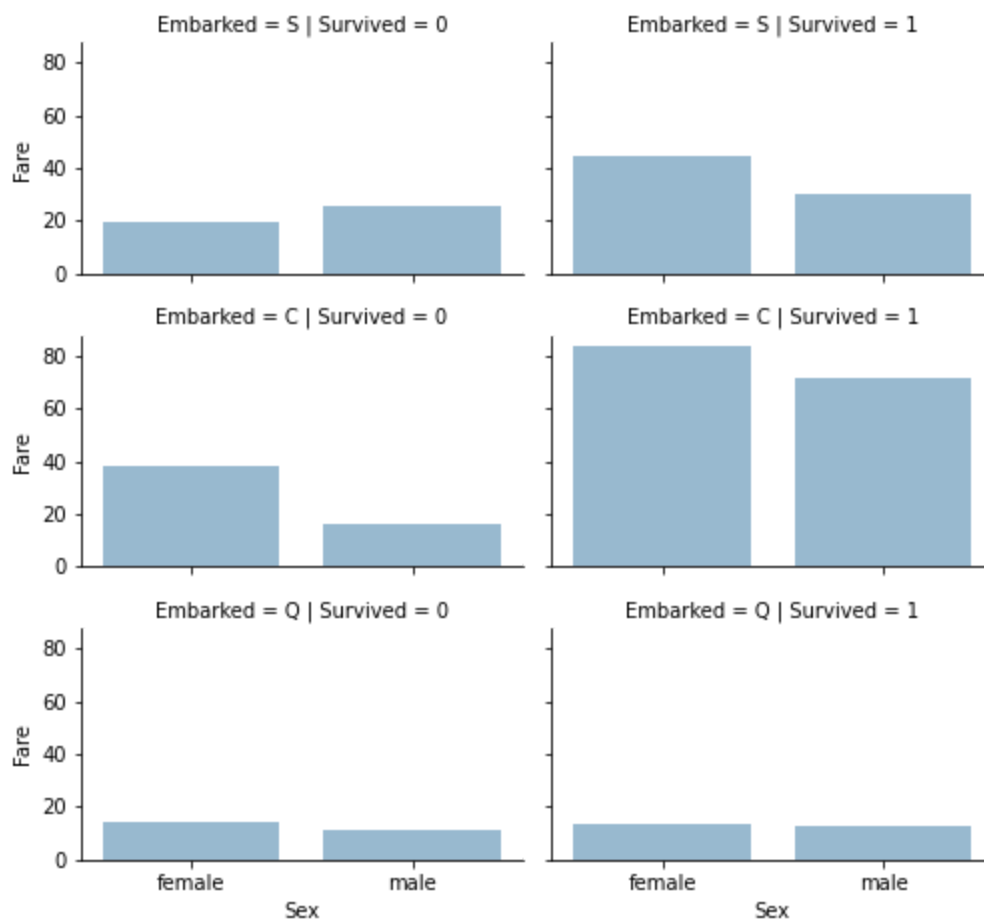
Out[18]: <seaborn.axisgrid.FacetGrid at 0x253ffd32c10>



```
In [19]: grid = sns.FacetGrid(train_df, row='Embarked', col='Survived', size=2.2, aspect=1.6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
```

```
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:316: UserWarning: The `size` parameter has been renamed to `height`; please update your
code.
  warnings.warn(msg, UserWarning)
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:643: UserWarning: Using the barplot function without specifying `order` is likely to pro
duce an incorrect plot.
  warnings.warn(warning)
```

```
Out[19]: <seaborn.axisgrid.FacetGrid at 0x253ffd82430>
```



```
In [20]: print("Before", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape)

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

"After", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape
```

Before (891, 12) (418, 11) (891, 12) (418, 11)

```
Out[20]: ('After', (891, 10), (418, 9), (891, 10), (418, 9))
```

```
In [21]: for dataset in combine:
          dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

pd.crosstab(train_df['Title'], train_df['Sex'])
```

```
Out[21]:
```

	Sex female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Maio	0	2

	Sex	female	male
Title			
Master		0	40
Miss		182	0
Mlle		2	0
Mme		1	0
Mr		0	517
Mrs		125	0
Ms		1	0
Rev		0	6
Sir		0	1

```
In [23]: for dataset in combine:
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')

train_df[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()
```

```
Out[23]:
```

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Rare	0.347826

```
In [24]: title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for dataset in combine:
dataset['Title'] = dataset['Title'].map(title_mapping)
dataset['Title'] = dataset['Title'].fillna(0)

train_df.head()
```

```
Out[24]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	3
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	2

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	Title	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	3
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1

```
In [25]: train_df = train_df.drop(['Name', 'PassengerId'], axis=1)
test_df = test_df.drop(['Name'], axis=1)
combine = [train_df, test_df]
train_df.shape, test_df.shape
```

```
Out[25]: ((891, 9), (418, 9))
```

```
In [26]: for dataset in combine:
dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)

train_df.head()
```

```
Out[26]:
```

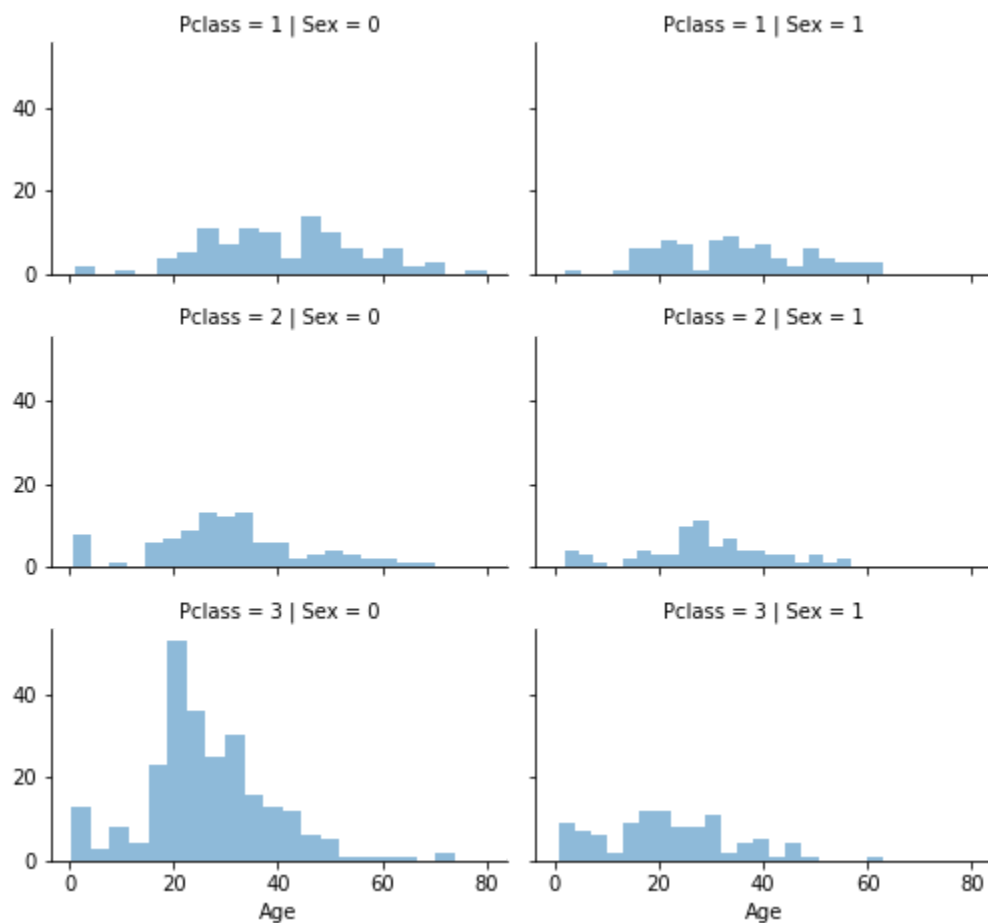
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	S	1
1	1	1	1	38.0	1	0	71.2833	C	3
2	1	3	1	26.0	0	0	7.9250	S	2
3	1	1	1	35.0	1	0	53.1000	S	3
4	0	3	0	35.0	0	0	8.0500	S	1

```
In [27]: # grid = sns.FacetGrid(train_df, col='Pclass', hue='Gender')
grid = sns.FacetGrid(train_df, row='Pclass', col='Sex', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
```

```
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
y:316: UserWarning: The `size` parameter has been renamed to `height`; please update your
code.
warnings.warn(msg, UserWarning)
```

```
Out[27]: <seaborn.axisgrid.FacetGrid at 0x253830df250>
```





```
In [28]: guess_ages = np.zeros((2,3))
guess_ages
```

```
Out[28]: array([[0., 0., 0.],
               [0., 0., 0.]])
```

```
In [29]: for dataset in combine:
          for i in range(0, 2):
              for j in range(0, 3):
                  guess_df = dataset[(dataset['Sex'] == i) & \
                                      (dataset['Pclass'] == j+1)]['Age'].dropna()

                  # age_mean = guess_df.mean()
                  # age_std = guess_df.std()
                  # age_guess = rnd.uniform(age_mean - age_std, age_mean + age_std)

                  age_guess = guess_df.median()

                  # Convert random age float to nearest .5 age
                  guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

          for i in range(0, 2):
              for j in range(0, 3):
                  dataset.loc[ (dataset.Age.isnull()) & (dataset.Sex == i) & (dataset.Pclass ==
                      'Age'] = guess_ages[i,j]

          dataset['Age'] = dataset['Age'].astype(int)

train_df.head()
```

```
Out[29]:
```

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	3	0	22	1	0	7.2500	S	1

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
1	1	1	1	38	1	0	71.2833	C	3
2	1	3	1	26	0	0	7.9250	S	2
3	1	1	1	35	1	0	53.1000	S	3
4	0	3	0	35	0	0	8.0500	S	1

```
In [30]: train_df['AgeBand'] = pd.cut(train_df['Age'], 5)
train_df[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values
```

```
Out[30]:
```

	AgeBand	Survived
0	(-0.08, 16.0]	0.550000
1	(16.0, 32.0]	0.337374
2	(32.0, 48.0]	0.412037
3	(48.0, 64.0]	0.434783
4	(64.0, 80.0]	0.090909

```
In [31]: for dataset in combine:
dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
dataset.loc[ dataset['Age'] > 64, 'Age']
train_df.head()
```

```
Out[31]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title	AgeBand
0	0	3	0	1	1	0	7.2500	S	1	(16.0, 32.0]
1	1	1	1	2	1	0	71.2833	C	3	(32.0, 48.0]
2	1	3	1	1	0	0	7.9250	S	2	(16.0, 32.0]
3	1	1	1	2	1	0	53.1000	S	3	(32.0, 48.0]
4	0	3	0	2	0	0	8.0500	S	1	(32.0, 48.0]

```
In [32]: train_df = train_df.drop(['AgeBand'], axis=1)
combine = [train_df, test_df]
train_df.head()
```

```
Out[32]:
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	1	1	0	7.2500	S	1
1	1	1	1	2	1	0	71.2833	C	3
2	1	3	1	1	0	0	7.9250	S	2
3	1	1	1	2	1	0	53.1000	S	3
4	0	3	0	2	0	0	8.0500	S	1

```
In [33]: for dataset in combine:
dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
```

```
train_df[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean().sort_
```

Out[33]:

	FamilySize	Survived
3	4	0.724138
2	3	0.578431
1	2	0.552795
6	7	0.333333
0	1	0.303538
4	5	0.200000
5	6	0.136364
7	8	0.000000
8	11	0.000000

In [34]:

```
for dataset in combine:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1

train_df[['IsAlone', 'Survived']].groupby(['IsAlone'], as_index=False).mean()
```

Out[34]:

	IsAlone	Survived
0	0	0.505650
1	1	0.303538

In [35]:

```
train_df = train_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
test_df = test_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
combine = [train_df, test_df]

train_df.head()
```

Out[35]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone
0	0	3	0	1	7.2500	S	1	0
1	1	1	1	2	71.2833	C	3	0
2	1	3	1	1	7.9250	S	2	1
3	1	1	1	2	53.1000	S	3	0
4	0	3	0	2	8.0500	S	1	1

In [36]:

```
for dataset in combine:
    dataset['Age*Class'] = dataset.Age * dataset.Pclass

train_df.loc[:, ['Age*Class', 'Age', 'Pclass']].head(10)
```

Out[36]:

	Age*Class	Age	Pclass
0	3	1	3
1	2	2	1
2	3	1	3

	Age*Class	Age	Pclass
3	2	2	1
4	6	2	3
5	3	1	3
6	3	3	1
7	0	0	3
8	3	1	3
9	0	0	2

```
In [37]: freq_port = train_df.Embarked.dropna().mode()[0]
freq_port
```

Out[37]: 'S'

```
In [38]: for dataset in combine:
dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)

train_df[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean().sort_valu
```

```
Out[38]:
```

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

```
In [39]: for dataset in combine:
dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

train_df.head()
```

```
Out[39]:
```

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	7.2500	0	1	0	3
1	1	1	1	2	71.2833	1	3	0	2
2	1	3	1	1	7.9250	0	2	1	3
3	1	1	1	2	53.1000	0	3	0	2
4	0	3	0	2	8.0500	0	1	1	6

```
In [40]: test_df['Fare'].fillna(test_df['Fare'].dropna().median(), inplace=True)
test_df.head()
```

```
Out[40]:
```

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	892	3	0	2	7.8292	2	1	1	6
1	893	3	1	2	7.0000	0	3	0	6
2	894	2	0	3	9.6875	2	1	1	6
3	895	3	0	1	8.6625	0	1	1	3

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
4	896	3	1	1	12.2875	0	3	0	3

```
In [41]: train_df['FareBand'] = pd.qcut(train_df['Fare'], 4)
train_df[['FareBand', 'Survived']].groupby(['FareBand'], as_index=False).mean().sort_valu
```

```
Out[41]:
```

	FareBand	Survived
0	(-0.001, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31.0]	0.454955
3	(31.0, 512.329]	0.581081

```
In [42]: for dataset in combine:
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, 'Fare'] = 3
dataset['Fare'] = dataset['Fare'].astype(int)

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

```
In [43]: train_df.head(10)
```

```
Out[43]:
```

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	0	0	1	0	3
1	1	1	1	2	3	1	3	0	2
2	1	3	1	1	1	0	2	1	3
3	1	1	1	2	3	0	3	0	2
4	0	3	0	2	1	0	1	1	6
5	0	3	0	1	1	2	1	1	3
6	0	1	0	3	3	0	1	1	3
7	0	3	0	0	2	0	4	0	0
8	1	3	1	1	1	0	3	0	3
9	1	2	1	0	2	1	3	0	0

Model, predict and solve Now we are ready to train a model and predict the required solution. There are 60+ predictive modelling algorithms to choose from. We must understand the type of problem and solution requirement to narrow down to a select few models which we can evaluate. Our problem is a classification and regression problem. We want to identify relationship between output (Survived or not) with other variables or features (Gender, Age, Port...). We are also performing a category of machine learning which is called supervised learning as we are training our model with a given dataset. With these two criteria - Supervised Learning plus Classification and Regression, we can narrow down our choice of models to a few. These include:

Logistic Regression KNN or k-Nearest Neighbors Support Vector Machines Naive Bayes classifier  
Decision Tree Random Forrest Perceptron Artificial neural network RVM or Relevance Vector  
Machine

```
In [44]: X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test = test_df.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

```
Out[44]: ((891, 8), (891,), (418, 8))
```

```
In [45]: # 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)
acc_log
```

```
Out[45]: 80.36
```

```
In [46]: coeff_df = pd.DataFrame(train_df.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

```
Out[46]:
```

	Feature	Correlation
1	Sex	2.201619
5	Title	0.397888
2	Age	0.287011
4	Embarked	0.261473
6	IsAlone	0.126553
3	Fare	-0.086655
7	Age*Class	-0.311069
0	Pclass	-0.750700

```
In [47]: # Support Vector Machines

svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

```
Out[47]: 78.23
```

```
In [48]: 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)
```

Out[48]: 84.74

In [49]:

```
# 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)
acc_gaussian
```

Out[49]: 72.28

In [50]:

```
# Perceptron

perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron
```

Out[50]: 78.34

In [51]:

```
# 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)
acc_linear_svc
```

```
c:\users\andre\appdata\local\programs\python\python39\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

Out[51]: 79.12

In [52]:

```
# Stochastic Gradient Descent

sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd
```

Out[52]: 77.44

In [53]:

```
# 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)
acc_decision_tree
```

Out[53]: 86.76

```
In [54]: # Random Forest

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = 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)
acc_random_forest
```

Out[54]: 86.76

```
In [55]: models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

Out[55]:

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
2	Logistic Regression	80.36
7	Linear SVC	79.12
5	Perceptron	78.34
0	Support Vector Machines	78.23
6	Stochastic Gradient Decent	77.44
4	Naive Bayes	72.28

```
In [56]: submission = pd.DataFrame({
    "PassengerId": test_df["PassengerId"],
    "Survived": Y_pred
})
```

In [ ]: