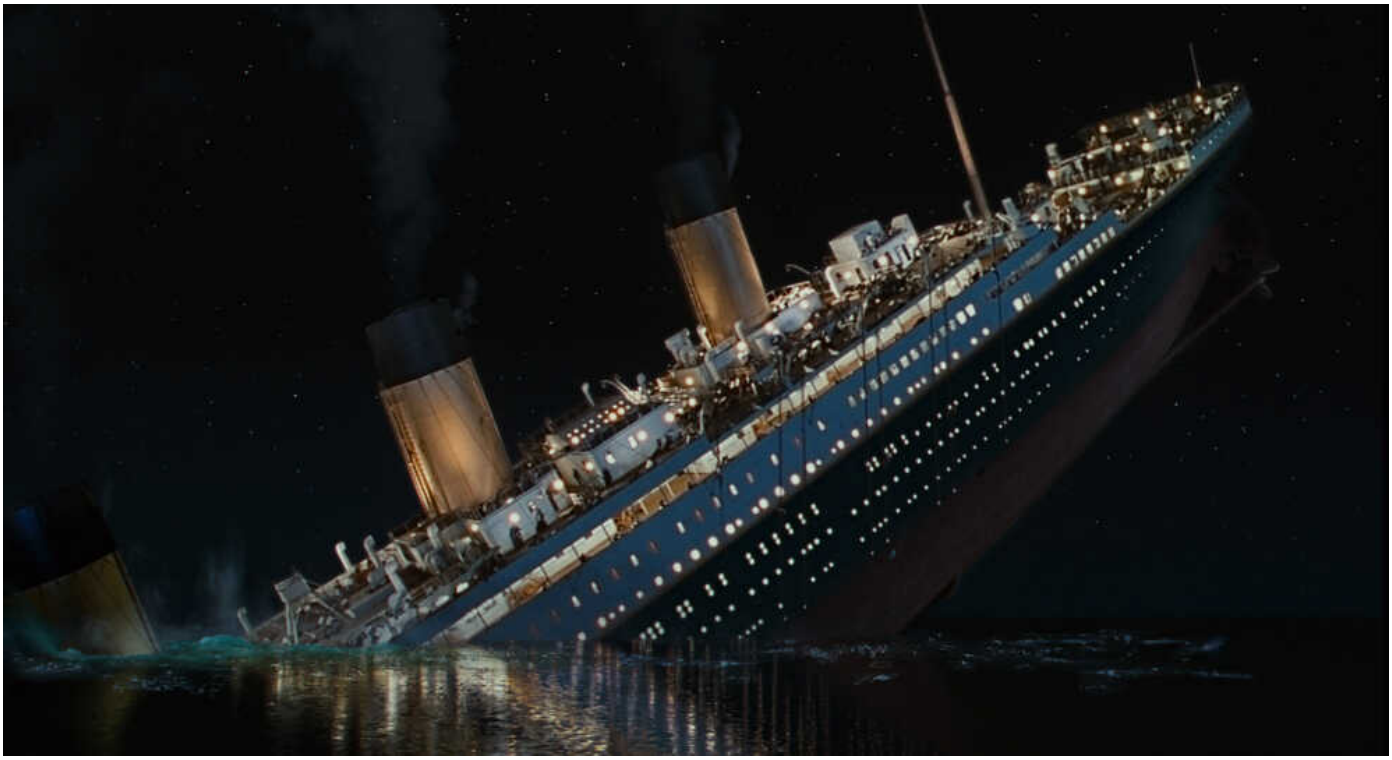
Titanic surviaval rate prediction



Some important **import** statements

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

Cleaning Features

Treat Missing Values In The Data

Read In Data

In [ ]:

```
titanic = pd.read_csv('train.csv')
titanic.head()
```

Out[118]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Our dataset contain several features including:

- **Name** (str) - Name of the passenger
- **Pclass** (int) - Ticket class (1st, 2nd, or 3rd)
- **Sex** (str) - Gender of the passenger
- **Age** (float) - Age in years
- **SibSp** (int) - Number of siblings and spouses aboard
- **Parch** (int) - Number of parents and children aboard
- **Ticket** (str) - Ticket number
- **Fare** (float) - Passenger fare
- **Cabin** (str) - Cabin number
- **Embarked** (str) - Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

Check for missing values

In [ ]:

```
titanic.isnull().sum()
```

Out[6]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Fill missing for `Age`

First check if Age is missing at random using the `groupby()` method

Fill the missing values for `Age` with the mean

In [ ]:

```
titanic['Age_clean']= titanic['Age'].fillna(titanic['Age'].mean())
titanic.head()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Take a look on your data

In [ ]:

```
titanic.isnull().sum()
```

Fill Missing For `Embarked`

Note that the value for `Embarked` is char so we will replace it with a letter to refer that it's missing

In [ ]:

```
titanic['Embarked_clean'] = titanic['Embarked'].fillna('X')
```

Take a look on your data

In [ ]:

```
titanic.head()
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [ ]:

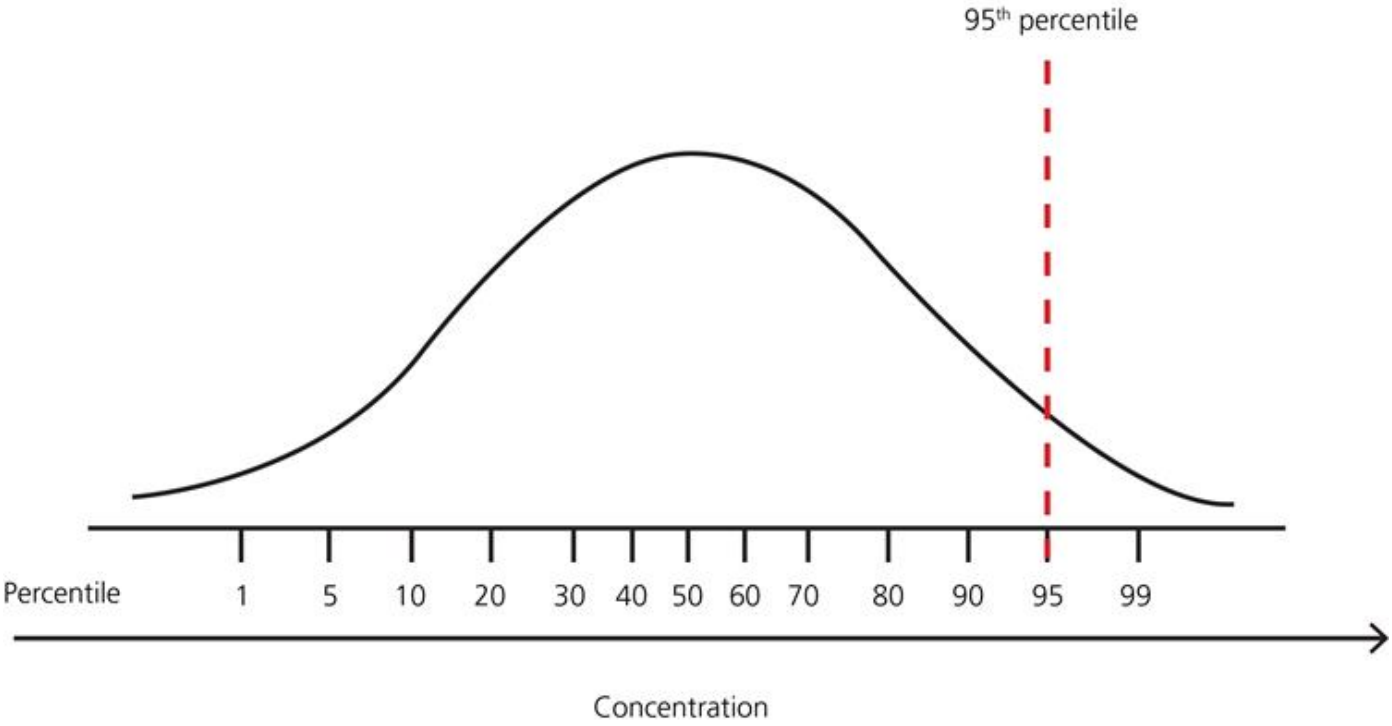
```
titanic.isnull().sum()
```

Out[13]:

```
PassengerId         0
Survived            0
Pclass              0
Name                0
Sex                 0
Age               177
SibSp               0
Parch               0
Ticket              0
Fare                0
Cabin             687
Embarked            2
Age_clean           0
Embarked_clean      0
dtype: int64
```

Cap And Floor Data To Remove Outliers

This function uses the `quantile` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.quantile.html) function from `pandas` to detect outliers above the 95 percentile



In [ ]:

```
def detect_outlier(feature , dataFrame):
    outliers = []
    data = dataFrame[feature]
    mean = np.mean(data)
    std =np.std(data)


    for y in data:
        z_score= (y - mean)/std
        if np.abs(z_score) > 3:
            outliers.append(y)
    print('\nOutlier caps for {}:'.format(feature))
    print('  --95p: {:.1f} / {} values exceed that'.format(data.quantile(.99),
                                                           len([i for i in data
                                                                if i > data.quantile(.99)
```

See the description of your data

In [ ]:

```
titanic.describe()
```

Out[15]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	A
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000	8
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	

Use the 95 percentile as an upperbound (if needed) for continuous features and cap the values using the `clip` (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.clip.html) function from pandas

In [ ]:

```
for feature in ['SibSp' , 'Parch' , 'Fare' , 'Age_clean']:
    detect_outlier(feature,titanic)
```

```
Outlier caps for SibSp:
  --95p: 5.0 / 7 values exceed that

Outlier caps for Parch:
  --95p: 4.0 / 6 values exceed that

Outlier caps for Fare:
  --95p: 249.0 / 9 values exceed that

Outlier caps for Age_clean:
  --95p: 65.0 / 8 values exceed that
```

In [ ]:

```
titanic['Fare_clean'] = titanic['Fare'].clip(upper = titanic['Fare'].quantile(0.99))
titanic['Age_clean'].clip(upper = titanic['Age_clean'].quantile(0.99) , inplace = True)
```

See the description of your data after capping

In [ ]:

```
titanic.describe()
```

Out[29]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	A
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000	8
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	

Create New Features From Text

Create a title feature by parsing passenger name

using the `lambda` (https://www.youtube.com/watch?list=PL98qAXLA6aftqPGddFjJ59m71F96ZPwD4&time_continue=79&v=dX6lWSp7pP4&feature=emb_logo), `split()` (https://pandas.pydata.org/docs/reference/api/pandas.Series.str.split.html?

highlight=split#pandas.Series.str.split), and `strip()` (https://pandas.pydata.org/docs/reference/api/pandas.Series.str.strip.html?highlight=strip#pandas.Series.str.strip)

In [ ]:

```
titanic['Title'] = titanic['Name'].apply(lambda x : x.split(',')[1].split('.')[0].strip())
titanic.head()
```

Out[30]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Look at survival rate by `Title` and `Sex` using a `pivot_table` (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pivot_table.html?highlight=pivot_table#pandas.DataFrame.pivot_table)

In [ ]:

```
titanic.pivot_table('Survived' , index = ['Title', 'Sex'] , aggfunc = ['mean'])
```

Out[31]:

		mean
		Survived
Title	Sex	
Capt	male	0.000000
Col	male	0.500000
Don	male	0.000000
Dr	female	1.000000
	male	0.333333
Jonkheer	male	0.000000
Lady	female	1.000000
Major	male	0.500000
Master	male	0.575000
Miss	female	0.697802
Mlle	female	1.000000
Mme	female	1.000000
Mr	male	0.156673
Mrs	female	0.792000
Ms	female	1.000000
Rev	male	0.000000
Sir	male	1.000000
the Countess	female	1.000000

Create Indicator

Create Indicator Variable For `Cabin`

Using the `where()` (https://numpy.org/doc/stable/reference/generated/numpy.where.html) function from `numpy` we will assign numeric values for `Cabin`

- If `Cabin` is null this means that this passenger didn't have a cabin this will be represented by **0**
- If `Cabin` has a value this will be represented by **1**

In [ ]:

```
titanic['Cabin_clean'] = np.where(titanic['Cabin'].isnull() , 0 , 1)
```

Take a look on your data

In [ ]:

```
titanic.head()
```

Out[36]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Combine Existing Features Into New Feature

Combine `SibSp` & `Parch` Into New `Family` Feature

In [ ]:

```
titanic['Family'] = titanic['SibSp']+titanic['Parch']
titanic.head()
```

Out[37]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Convert Categorical Features To Numeric

Categorical Features that we have include:

`Sex` , `Embarked` , `Embarked_clean` , `Title` , `Cabin`

We will convert the features using the `fit_transform` (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder.fit_transform) method from `LabelEncoder`

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
for feature in ['Sex' , 'Embarked' , 'Embarked_clean' , 'Title' , 'Cabin']: # type the name
    le = LabelEncoder()
    titanic[feature] = le.fit_transform(titanic[feature].astype(str))
```

In [ ]:

```
titanic.head()
```

Out[49]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cal
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	1

Prepare Features For Modeling

Check the correlation between the features and the target

In [ ]:

```
titanic.corr()
```

Out[43]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch
PassengerId	1.000000	-0.005007	-0.035144	0.042939	0.036847	-0.057527	-0.001652
Survived	-0.005007	1.000000	-0.338481	-0.543351	-0.077221	-0.035322	0.081629
Pclass	-0.035144	-0.338481	1.000000	0.131900	-0.369226	0.083081	0.018443
Sex	0.042939	-0.543351	0.131900	1.000000	0.093254	-0.114631	-0.245489
Age	0.036847	-0.077221	-0.369226	0.093254	1.000000	-0.308247	-0.189119
SibSp	-0.057527	-0.035322	0.083081	-0.114631	-0.308247	1.000000	0.414838
Parch	-0.001652	0.081629	0.018443	-0.245489	-0.189119	0.414838	1.000000
Fare	0.012658	0.257307	-0.549500	-0.182333	0.096067	0.159651	0.216225
Embarked	0.013083	-0.163517	0.157112	0.104057	-0.025252	0.066654	0.038322
Age_clean	0.031961	-0.069868	-0.332494	0.081952	0.998564	-0.234131	-0.180092
Embarked_clean	0.013083	-0.163517	0.157112	0.104057	-0.025252	0.066654	0.038322
Fare_clean	0.006677	0.273008	-0.606050	-0.212492	0.104818	0.192869	0.248623
Title	0.035120	-0.193635	0.029099	0.250075	0.307794	-0.200046	-0.126422
Cabin_clean	0.019919	0.316912	-0.725541	-0.140391	0.249732	-0.040460	0.036987
Family	-0.040143	0.016639	0.065997	-0.200988	-0.301914	0.890712	0.783111

Using original data

Create Training And Test Sets

Choose your **features** and **target**

In [ ]:

```
f = ['Age_clean' , 'Sex' , 'Fare' , 'Pclass' , 'Parch' , 'SibSp' , 'Cabin', 'Embarked']
features = titanic [f] # The list of features
target = titanic['Survived']
```

Split the data using the `train_test_split` (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train_test_split#sklearn.model_selection.train_test_split) method from `sklearn`

In [ ]:

```
X_train , X_test , y_train , y_test = train_test_split(features , target , train_size = 0.6
X_test , X_val , y_test , y_val = train_test_split(X_test , y_test , train_size = 0.5 , ran
```

Build a `Random Forest` (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random+forest+classifier) model using `RandomForestClassifier` from `sklearn`

In [ ]:

```
Random_forest_model = RandomForestClassifier(random_state = 12)
Random_forest_model.fit(X_train , y_train)
```

Out[68]:

```
RandomForestClassifier(random_state=12)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

evaluate the model using this code

In [ ]:

```
from sklearn.metrics import accuracy_score
def evaluate_model(model, features, labels):
    pred = model.predict(features)
    accuracy = round(accuracy_score(labels, pred), 3)
    print('-- \tAccuracy: {} '.format(accuracy))
```

In [ ]:

```
evaluate_model(Random_forest_model , X_test , y_test)
```

```
--      Accuracy: 0.758
```

Tune hyperparameters using `GridSearchCV` (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=gridsearchcv#sklearn.model_selection.GridSearchCV)

We will tune 2 hyperparameters

- `max_depth`
- `n_estimators`

In [ ]:

```
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))
```

In [ ]:

```
parameters = {'max_depth':[3, 5 , 8],
              'n_estimators': [100 , 200, 300]}
t_model = GridSearchCV(Random_forest_model , parameters)
t_model.fit(X_train , y_train)
print_results(t_model)
```

```
BEST PARAMS: {'max_depth': 5, 'n_estimators': 300}

0.815 (+/-0.06) for {'max_depth': 3, 'n_estimators': 100}
0.809 (+/-0.05) for {'max_depth': 3, 'n_estimators': 200}
0.807 (+/-0.046) for {'max_depth': 3, 'n_estimators': 300}
0.833 (+/-0.043) for {'max_depth': 5, 'n_estimators': 100}
0.835 (+/-0.043) for {'max_depth': 5, 'n_estimators': 200}
0.841 (+/-0.035) for {'max_depth': 5, 'n_estimators': 300}
0.839 (+/-0.077) for {'max_depth': 8, 'n_estimators': 100}
0.831 (+/-0.052) for {'max_depth': 8, 'n_estimators': 200}
0.837 (+/-0.049) for {'max_depth': 8, 'n_estimators': 300}
```

Using cleaned data

Create Training And Test Sets

Choose your **features** and **target**

In [ ]:

```
cf = ['Age_clean' , 'Pclass' ,'Sex', 'Cabin_clean' , 'Family' , 'Title' ,'Fare_clean']
c_features = titanic [cf] # The list of features
target = titanic['Survived']
```

Split the data using the `train_test_split` (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train_test_split#sklearn.model_selection.train_test_split) method from `sklearn`

In [ ]:

```
X1_train , X1_test , y1_train , y1_test = train_test_split(c_features , target , train_size
X1_test , X1_val , y1_test , y1_val = train_test_split(X1_test , y1_test , train_size = 0.5
```

Build a `Random Forest` (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?

highlight=random+forest+classifier) model using `RandomForestClassifier` from `sklearn`

In [ ]:

```
c_model = RandomForestClassifier(random_state = 12)
c_model.fit(X1_train , y1_train)
```

Out[86]:

```
RandomForestClassifier(random_state=12)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

evaluate the model

In [ ]:

```
evaluate_model(c_model , X1_test , y1_test)
```

```
--      Accuracy: 0.798
```

In [ ]:

```
grid_model = GridSearchCV(c_model , parameters)
grid_model.fit(X1_train , y1_train)
print_results(grid_model)
```

```
BEST PARAMS: {'max_depth': 5, 'n_estimators': 200}

0.848 (+/-0.022) for {'max_depth': 3, 'n_estimators': 100}
0.85 (+/-0.029) for {'max_depth': 3, 'n_estimators': 200}
0.848 (+/-0.03) for {'max_depth': 3, 'n_estimators': 300}
0.854 (+/-0.035) for {'max_depth': 5, 'n_estimators': 100}
0.86 (+/-0.027) for {'max_depth': 5, 'n_estimators': 200}
0.856 (+/-0.019) for {'max_depth': 5, 'n_estimators': 300}
0.848 (+/-0.018) for {'max_depth': 8, 'n_estimators': 100}
0.843 (+/-0.03) for {'max_depth': 8, 'n_estimators': 200}
0.846 (+/-0.027) for {'max_depth': 8, 'n_estimators': 300}
```

If you would like to practice more you can use this link (https://drive.google.com/drive/folders/1JiTjMGVjC476NuywW5hP0n7WWCaQZ1__?usp=sharing)

you will find another dataset that needs the same process and a file to describe its features