In [1]:

```
import pandas as pd
import numpy as np
import random as rnd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
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
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder, LabelEncod
er
from sklearn.model_selection import train_test_split, KFold, cross_val_score, cross_val_p
redict, cross_validate, GridSearchCV
import xgboost as xgb


train_dataset = pd.read_csv('train.csv')
test_dataset = pd.read_csv('test.csv')
combine = [train_dataset, test_dataset]
```

In [2]:

```
# train_dataset['isCabinNull'] = train_dataset['Cabin'].isnull()*1
# train_dataset.head()
```

In [3]:

```
# test_dataset['isCabinNull'] = test_dataset['Cabin'].isnull()*1
# test_dataset.head()
```

In [4]:

```
train_dataset=train_dataset.drop("Ticket",axis=1)
train_dataset=train_dataset.drop("Cabin",axis=1)

train_dataset.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S

In [5]:

```
test_dataset=test_dataset.drop("Ticket",axis=1)
test_dataset=test_dataset.drop("Cabin",axis=1)

test_dataset.head()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	7.8292	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	7.0000	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	9.6875	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	8.6625	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	12.2875	S

In [6]:

```
train_dataset['FamilySize'] = train_dataset['SibSp'] + train_dataset['Parch'] + 1

train_dataset.head()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1

In [7]:

```
test_dataset['FamilySize'] = test_dataset['SibSp'] + test_dataset['Parch'] + 1

train_dataset.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	2
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1

In [8]:

```
def categorise(row):
    if row > 1 and row < 5:
        return 1
    return 0
train_dataset['NormalFamilySize'] = train_dataset['FamilySize'].apply(categorise)

train_dataset.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	2	1

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	2	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	1	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	2	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1	0

In [9]:

```
def categorise(row):
    if row > 1 and row < 5:
        return 1
    return 0
test_dataset['NormalFamilySize'] = test_dataset['FamilySize'].apply(categorise)

test_dataset.head()
```

Out[9]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
0	892	3	Kelly, Mr. James	male	34.5	0	0	7.8292	Q	1	0
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	7.0000	S	2	1
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	9.6875	Q	1	0
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	8.6625	S	1	0
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	12.2875	S	3	1

In [10]:

```
data = [train_dataset, test_dataset]

for each in data:
    meanAge = train_dataset["Age"].mean()
    stdAge = test_dataset["Age"].std()
    isNull = each["Age"].isnull().sum()
    # print(isNull)
    # rand_age = np.random.randint(meanAge - stdAge, meanAge + stdAge)
    randomAgeGenerator = np.random.randint(meanAge - stdAge, meanAge + stdAge, size = is
Null)
    print(randomAgeGenerator)
    age_copy = each["Age"].copy()
    age_copy[np.isnan(age_copy)] = randomAgeGenerator
    each["Age"] = age_copy
    each["Age"] = train_dataset["Age"].astype(int)
```

```
[34 36 15 37 16 18 28 30 28 25 40 34 37 21 30 20 27 42 20 23 41 28 31 36
 17 32 23 38 42 36 25 18 42 29 23 24 36 20 16 21 23 16 22 40 37 31 37 41
 34 16 21 31 31 33 28 19 19 31 25 26 16 37 42 17 29 18 41 35 35 25 37 40
 23 24 33 31 19 20 16 21 27 32 40 29 27 41 21 27 42 30 15 16 17 36 24 16
 33 27 32 37 24 16 26 18 33 37 41 29 39 19 41 23 34 21 40 38 35 39 26 29
 37 21 19 37 21 21 26 31 21 33 38 40 22 31 35 31 26 41 22 25 28 20 39 41
 19 23 28 33 15 17 15 17 15 24 23 37 16 34 29 28 21 38 18 26 42 29 27 27
 28 40 38 18 34 33 28 19 20]
```

```
[17 35 19 37 20 42 36 23 32 20 21 37 34 31 32 36 18 24 24 35 32 30 38 33
 27 27 21 26 30 27 18 19 24 36 40 33 34 28 28 36 38 38 21 30 17 38 37 28
 28 15 33 29 30 30 38 15 16 22 36 26 25 22 32 28 41 36 21 36 37 37 36 37
 21 28 23 37 23 39 32 25 20 36 16 27 24 18]
```

In [11]:

```
# train_dataset["Embarked"] = train_dataset["Embarked"].fillna('C')
train_dataset["Embarked"] = train_dataset["Embarked"].fillna('S')

train_dataset.head()
```

Out[11]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	7.2500	S	2	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	71.2833	C	2	1
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	7.9250	S	1	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	53.1000	S	2	1
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	8.0500	S	1	0

In [12]:

```
test_dataset = test_dataset.fillna(test_dataset['Fare'].mean())
```

In [13]:

```
train_dataset=train_dataset.drop("PassengerId",axis=1)
train_dataset=train_dataset.drop("Name",axis=1)

train_dataset.head()
```

Out[13]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
0	0	3	male	22	1	0	7.2500	S	2	1
1	1	1	female	38	1	0	71.2833	C	2	1
2	1	3	female	26	0	0	7.9250	S	1	0
3	1	1	female	35	1	0	53.1000	S	2	1
4	0	3	male	35	0	0	8.0500	S	1	0

In [14]:

```
test_dataset=test_dataset.drop("PassengerId",axis=1)
test_dataset=test_dataset.drop("Name",axis=1)

test_dataset.head()
```

Out[14]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	NormalFamilySize
0	3	male	22	0	0	7.8292	Q	1	0
1	3	female	38	1	0	7.0000	S	2	1
2	2	male	26	0	0	9.6875	Q	1	0
3	3	male	35	0	0	8.6625	S	1	0
4	3	female	35	1	1	12.2875	S	3	1

In [15]:

```
le = LabelEncoder()
train_dataset["Sex"]= le.fit_transform(train_dataset["Sex"])
print(train_dataset["Sex"])
```

```
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
890    1
Name: Sex, Length: 891, dtype: int64
```

In [16]:

```
test_dataset["Sex"]= le.fit_transform(test_dataset["Sex"])
print(test_dataset["Sex"])
```

```
0      1
1      0
2      1
3      1
4      0
      ..
413    1
414    0
415    1
416    1
417    1
Name: Sex, Length: 418, dtype: int64
```

In [17]:

```
train_dataset["Embarked"]= le.fit_transform(train_dataset["Embarked"])
print(train_dataset["Embarked"])
```

```
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 891, dtype: int64
```

In [18]:

```
test_dataset["Embarked"]= le.fit_transform(test_dataset["Embarked"])
print(test_dataset["Embarked"])
```

```
0      1
```

```
1      2
2      1
3      2
4      2
      ..
413    2
414    0
415    2
416    2
417    0
Name: Embarked, Length: 418, dtype: int64
```

In [19]:

```
X_train = train_dataset.drop("Survived", axis=1)
Y_train = train_dataset["Survived"]
X_test = test_dataset.copy()
```

In [20]:

```
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [21]:

```
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
log_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

Out[21]:

```
81.14
```

In [22]:

```
knn = KNeighborsClassifier(algorithm='auto', leaf_size=16, metric='minkowski',
                           metric_params=None, n_jobs=1, n_neighbors=6, p=2,
                           weights='uniform')
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

Out[22]:

```
84.06
```

In [23]:

```
svc = SVC(probability = True, kernel = 'rbf', random_state = 0)
svc.fit(X_train, Y_train)
svc_pred = svc.predict(X_test)
svc.score(X_train, Y_train)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

Out[23]:

```
83.28
```

In [24]:

```
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
gaussian_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

Out[24]:

```
78.68
```

In [25]:

```
# decision_tree = DecisionTreeClassifier(max_depth = 6)
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
dt_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

Out[25]:

```
98.2
```

In [26]:

```
# random_forest = RandomForestClassifier(max_depth=11, n_estimators=50, n_jobs=-1, random
_state=13)
random_forest = RandomForestClassifier()
random_forest.fit(X_train, Y_train)
rf_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[26]:

```
98.2
```

In [27]:

```
xgb_model = xgb.XGBClassifier()

xgb_model.fit(X_train, Y_train)
xgb_pred = xgb_model.predict(X_test)

acc_xgb = round(xgb_model.score(X_train, Y_train) * 100, 2)
acc_xgb
```

Out[27]:

```
96.52
```

In [28]:

```
mlp_model = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(11, 11, 11), random_state=
1, max_iter = 1000)

mlp_model.fit(X_train, Y_train)
mlp_pred = xgb_model.predict(X_test)

acc_mlp = round(mlp_model.score(X_train, Y_train) * 100, 2)
acc_mlp
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/n
eural_network/_multilayer_perceptron.py:559: 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
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Out[28]:

```
93.15
```

In [29]:

```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'XGBoost',
              'Decision Tree', 'MLP'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_xgb, acc_decision_tree, acc_mlp]})
models.sort_values(by='Score', ascending=False)
```

Out[29]:

	Model	Score
3	Random Forest	98.20
6	Decision Tree	98.20
5	XGBoost	96.52
7	MLP	93.15
1	KNN	84.06
0	Support Vector Machines	83.28
2	Logistic Regression	81.14
4	Naive Bayes	78.68

In [30]:

```
sft_voting = VotingClassifier (estimators = [
    ('LogisticRegression', logreg),
    ('SVC', svc),
    ('KNN', knn),
    ('Gaussian', gaussian),
#     ('perceptron', perceptron),
#     ('Linear SVC', linear_svc),
    ('Decision Tree', decision_tree),
    ('RF', random_forest),
#     ('SGD', sgd),
    ('xgb_model', xgb_model),
    ('mlp_model', mlp_model)
], weights = [
    1,
    8,
    2,
    1,
    2,
    2,
    2,
    2
], voting = 'soft')

sft_voting.fit(X_train, Y_train)
soft_pred = sft_voting.predict(X_test)
acc_soft = round(sft_voting.score(X_train, Y_train) * 100, 2)
acc_soft
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/sklearn/n
eural_network/_multilayer_perceptron.py:559: 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
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Out[30]:

```
90.91
```

In [33]:

```
#experiment section
import statistics
```

```python
experiment_pred = []
for index in range(0,418):
    list_of_pred = [
        log_pred[index],
        knn_pred[index],
        svc_pred[index],
        gaussian_pred[index],
        dt_pred[index],
        rf_pred[index],
        xgb_pred[index],
        mlp_pred[index]
#         soft_pred[index]
    ]
#     list_of_pred.sort()
    median_pred = statistics.median(list_of_pred)
#     print(list_of_pred)
    if median_pred == 0.5:
        median_pred = knn_pred[index]
#         median_pred = svc_pred[index]
    print(median_pred)
    experiment_pred.append(int(median_pred))
```

```
0.0
0.0
0.0
0.0
1.0
0.0
1
1.0
1.0
0.0
0.0
0.0
1.0
0.0
1.0
1.0
0.0
0.0
0.0
1.0
1.0
0.0
1.0
0
1.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
0
0.0
1.0
0.0
1
1
0.0
0.0
0.0
0.0
0.0
1.0
1.0
0.0
1.0
0.0
1.0
1.0
```

```
1.0
0.0
1.0
1.0
0.0
0.0
0.0
0.0
0
1.0
0.0
0.0
0.0
1.0
0.0
1.0
1.0
0.0
0.0
1.0
1
0.0
1
1.0
1.0
0.0
0.0
1.0
0.0
1.0
0
1
0.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
1.0
1.0
1.0
0.0
0.0
0.0
0.0
1.0
1.0
0.0
```

```
1.0
0.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
0.0
0.0
1.0
0.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
0.0
1
0.0
0.0
1.0
0.0
0.0
1.0
1.0
0.0
1.0
0.0
0.0
1.0
0.0
0.0
1.0
1.0
0.0
0.0
1.0
0.0
0.0
1.0
1.0
0.0
1.0
1.0
0.0
1.0
1.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
```

```
0.0
0.0
0
0.0
0.0
0.0
1.0
0.0
0.0
1.0
0.0
1.0
1
0.0
1.0
0.0
0.0
0.0
0.0
1.0
0
0.0
1.0
0.0
1.0
0.0
1.0
1
1.0
0.0
1.0
1.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
0.0
0
0.0
0.0
0.0
1.0
1.0
1.0
1.0
0.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
0.0
0.0
```

```
0.0
0.0
0.0
0.0
0.0
0.0
1.0
1.0
0.0
1.0
0.0
0.0
0
0.0
1
0.0
1.0
0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
0.0
0.0
0.0
0.0
1.0
0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
1.0
0.0
0.0
0
0.0
0.0
0.0
0.0
1.0
1.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
0.0
1.0
0.0
0.0
0.0
0.0
```

```
0.0
0.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
1.0
0.0
1.0
1.0
0.0
0.0
0.0
0
0.0
1.0
0.0
0.0
0
0.0
1.0
1.0
0.0
1.0
0.0
0.0
0
1.0
0.0
0.0
1.0
0.0
0.0
1.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
1.0
0.0
1.0
0.0
0.0
0.0
0.0
0.0
1.0
0.0
0.0
0.0
1.0
0.0
1.0
0.0
0.0
1.0
0.0
1.0
0.0
0.0
0
0.0
1.0
1.0
1.0
```

```
1.0
1.0
1.0
0.0
1.0
0.0
0.0
0.0
```

In [34]:

```
submission = pd.read_csv('gender_submission.csv')
submission['Survived'] = experiment_pred
submission.to_csv('submission.csv', index=False)
```

In [ ]: