# KNN

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=7, p=2,

weights='uniform')

0.7369080502168653

{'n\_neighbors': 7, 'weights': 'distance'}

# Decision Tree

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=300,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

splitter='best')

0.7545051573951613

{'max\_depth': 100}

# Logistic Regression : 0.857

LogisticRegression(C=100, 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)

0.8571533720240617

{'C': 1, 'max\_iter': 100, 'solver': 'lbfgs'}

# Gradient Boosting Classifier

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance', max\_depth=5,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=300,

presort='auto', random\_state=None, subsample=1.0, verbose=0,

warm\_start=False)

Best score: 0.8399976018871484

Best params: {'max\_depth': 300}

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Pclass' 'Embarked' 'Parch']

# RandomForest : 0.858

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=3, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=200, n\_jobs=1,

oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

0.8581754351072274

{'max\_depth': 3, 'n\_estimators': 200}

feature\_importances = ['Sex' 'Fare' 'Cabin' 'Pclass' 'Age' 'SibSp' 'Embarked' 'Parch']

# SVC

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

0.8026530708162062

{'C': 5, 'gamma': 0.01}

# Stochastic Gradient Descent

SGDClassifier(alpha=0.1, average=False, class\_weight=None, epsilon=0.1,

eta0=0.0, fit\_intercept=True, l1\_ratio=0.15,

learning\_rate='optimal', loss='hinge', max\_iter=1000, n\_iter=None,

n\_jobs=1, penalty='l2', power\_t=0.5, random\_state=None,

shuffle=True, tol=None, verbose=0, warm\_start=False)

0.8308278142444236

{'alpha': 0.01, 'max\_iter': 1000}

# Neural Network

MLPClassifier(activation='relu', alpha=0.01, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(100,), learning\_rate='constant',

learning\_rate\_init=0.001, max\_iter=1000, momentum=0.9,

nesterovs\_momentum=True, power\_t=0.5, random\_state=None,

shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1,

verbose=False, warm\_start=False)

0.7533776716601814

{'alpha': 0.0001, 'max\_iter': 1000}

# Navie Bayes – GaussianNB

score = 0.7847533632286996

roc = 0.7707780725022103

GaussianNB(priors=None)

0.8200855514885222

{'priors': None}

# Dummy Classifier

score = 0.5112107623318386

roc = 0.46914235190097264

DummyClassifier(constant=None, random\_state=None, strategy='stratified')

0.5

{'strategy': 'most\_frequent'}

# KNN

for i in [1,3,5,7]:

knn = KNeighborsClassifier(n\_neighbors=i)

knn.fit(X\_train, y\_train)

score = knn.score(X\_test,y\_test)

yp=knn.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test,yp)

roc\_auc=auc(fpr,tpr)

print('k = ',i)

print('score = ',score)

print('roc = ',roc\_auc)

k = 1

score = 0.7174887892376681

roc = 0.6949039264828737

k = 3

score = 0.7354260089686099

roc = 0.7045530492898914

k = 5

score = 0.7309417040358744

roc = 0.7079782790309107

k = 7

score = 0.7040358744394619

roc = 0.6800334168755222

# Decision Tree

for i in [100,200,300]:

dt = DecisionTreeClassifier(max\_depth = i).fit(X\_train, y\_train)

dt.score(X\_test,y\_test)

yp = dt.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test,yp)

roc\_auc=auc(fpr,tpr)

print('Max\_depth = ',i)

print('score = ',score)

print('roc = ',roc\_auc)

Max\_depth = 100

score = 0.7040358744394619

roc = 0.783625730994152

Max\_depth = 200

score = 0.7040358744394619

roc = 0.7705513784461152

Max\_depth = 300

score = 0.7040358744394619

roc = 0.7780701754385965

# Logistic Regression

for i in [0.1,1,100]:

lr = LogisticRegression(C=i).fit(X\_train, y\_train)

lr.score(X\_test,y\_test)

print('score = ',score)

yp = lr.predict(X\_test)

fpr, tpr, \_ = roc\_curve(y\_test,yp)

roc\_auc=auc(fpr,tpr)

print('C = ',i)

print('score = ',score)

print('roc = ',roc\_auc)

C = 0.1

score = 0.972972972972973

roc = 0.7828320802005012

C = 1

score = 0.972972972972973

roc = 0.7862573099415205

C = 100

score = 0.972972972972973

roc = 0.7805346700083542

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=300,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

splitter='best')

0.7311797303226109

{'max\_depth': 200}

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'Embarked' 'Pclass' 'SibSp' 'Parch']

1. The probability that the classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example

# Gradient Boosting Classifier

print('features', X\_train.columns)

for e in [100,200,300]:

for m in [3,4,5]:

gbdt = GradientBoostingClassifier(n\_estimators=e, max\_depth=m).fit(X\_train, y\_train)

score = gbdt.score(X\_train, y\_train)

yp\_X = gbdt.predict(X\_train)

aucscore = roc\_auc\_score(y\_train,yp\_X)

yp\_Xt = gbdt.predict\_proba(X\_test)

print('n\_estimators:', e)

print('max\_depth:', m)

print('X\_train score = ',score)

print('feature\_importances = ',

X\_train.columns[np.argsort(-gbdt.feature\_importances\_)].values)

print('roc\_auc = ',aucscore)

# GridSearchCV

grid\_values = {'max\_depth':[100,200,300]}

grid\_clf\_search = GridSearchCV(gbdt, param\_grid=grid\_values, scoring= 'roc\_auc')

grid\_clf\_search.fit(X\_train, y\_train)

print(grid\_clf\_search.estimator)

print(grid\_clf\_search.best\_score\_)

print(grid\_clf\_search.best\_params\_)

print('feature\_importances = ',

X\_train.columns[np.argsort(-gbdt.feature\_importances\_)].values)

features Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')

n\_estimators: 100

max\_depth: 3

X\_train score = 0.918918918918919

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Pclass' 'Parch' 'Embarked']

roc\_auc = 0.9023749999999999

n\_estimators: 100

max\_depth: 4

X\_train score = 0.9474474474474475

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'Pclass' 'Embarked' 'SibSp' 'Parch']

roc\_auc = 0.9363846153846153

n\_estimators: 100

max\_depth: 5

X\_train score = 0.9669669669669669

feature\_importances = ['Fare' 'Age' 'Sex' 'Cabin' 'SibSp' 'Pclass' 'Embarked' 'Parch']

roc\_auc = 0.9575961538461538

n\_estimators: 200

max\_depth: 3

X\_train score = 0.93993993993994

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Pclass' 'Parch' 'Embarked']

roc\_auc = 0.9263846153846154

n\_estimators: 200

max\_depth: 4

X\_train score = 0.96996996996997

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Embarked' 'Parch' 'Pclass']

roc\_auc = 0.9631923076923077

n\_estimators: 200

max\_depth: 5

X\_train score = 0.987987987987988

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Embarked' 'Pclass' 'Parch']

roc\_auc = 0.9855961538461538

n\_estimators: 300

max\_depth: 3

X\_train score = 0.9564564564564565

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Parch' 'Embarked' 'Pclass']

roc\_auc = 0.9483846153846153

n\_estimators: 300

max\_depth: 4

X\_train score = 0.9834834834834835

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'Pclass' 'Embarked' 'SibSp' 'Parch']

roc\_auc = 0.9795961538461538

n\_estimators: 300

max\_depth: 5

X\_train score = 0.987987987987988

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Embarked' 'Pclass' 'Parch']

roc\_auc = 0.9855961538461538

GradientBoostingClassifier(criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='deviance', max\_depth=5,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=300,

presort='auto', random\_state=None, subsample=1.0, verbose=0,

warm\_start=False)

0.8083536642690368

{'max\_depth': 300}

feature\_importances = ['Fare' 'Age' 'Cabin' 'Sex' 'SibSp' 'Embarked' 'Pclass' 'Parch']