In [60]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [61]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import FunctionTransformer, PowerTransformer
from sklearn.compose import ColumnTransformer
```

In [62]:
```
import xgboost as xgb
```

In [63]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, Stack
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
```

In [64]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [65]:
```
df=pd.read_csv('train.csv')
```

In [66]:
```
df.head()
```

Out[66]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [67]:
```
#checking shape of data
df.shape
```

Out[67]:
```
(891, 12)
```

In [68]:
```
#checking whole description of data
df.describe()
```

Out[68]:

	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 [69]:
```
df['Survived'].value_counts()
```

Out[69]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [70]:
```
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
```

Preprocessing the train data

In [71]:
```
#checking for missing values
```

```
df.isnull().sum()

#So Age , Cabin and Embarked columns have missing values which should be taken care of
```

Out[71]:
```
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
```

In [72]:
```
df.drop(['Name','Cabin','SibSp','Parch','Ticket','PassengerId'],axis='columns', inplac
```

In [73]:
```
df.head()
```

Out[73]:

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

In [74]:
```
#now imputing missing values of Age column by its mean
df['Age']=df['Age'].fillna(df['Age'].mean())

df.head()
```

Out[74]:

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

In [75]:
```
df.isnull().sum()
```

Out[75]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    2
dtype: int64
```

In [76]:
```
df.drop(columns=['Sex','Embarked'],inplace=True)
```

In [77]:
```
df.head()
```

Out[77]:

	Survived	Pclass	Age	Fare
0	0	3	22.0	7.2500
1	1	1	38.0	71.2833
2	1	3	26.0	7.9250
3	1	1	35.0	53.1000
4	0	3	35.0	8.0500

In [78]:
```
df.corr()
```

Out[78]:

	Survived	Pclass	Age	Fare
Survived	1.000000	-0.338481	-0.069809	0.257307
Pclass	-0.338481	1.000000	-0.331339	-0.549500
Age	-0.069809	-0.331339	1.000000	0.091566
Fare	0.257307	-0.549500	0.091566	1.000000

In [79]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[79]:
```
<AxesSubplot:>
```

In [80]:
```
X=df.drop(columns=['Survived'])
y=df['Survived']
```

Model Ensembling

In [81]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=0)
```

First modeling without Gentic Algorthm search

In [112…
```
estimators3 = [
    ('rf', RandomForestClassifier(n_estimators=500, random_state=42, max_depth=30, mir
    ('knn', KNeighborsClassifier(n_neighbors=100)),
    ('gbdt',GradientBoostingClassifier(),
    ('kmc',KMeans(n_clusters=15, algorithm='DBSCAN')))
]
```

In [113…
```
clf3 = StackingClassifier(
    estimators=estimators3,
    final_estimator=xgb.XGBClassifier( ),
    cv=10
)
```

In [114…
```
clf3.fit(X_train,y_train)
```

Out[114]:
```
StackingClassifier(cv=10,
                   estimators=[('rf',
                                RandomForestClassifier(max_depth=30,
                                                       min_samples_leaf=7,
                                                       n_estimators=500,
                                                       random_state=42)),
                               ('knn', KNeighborsClassifier(n_neighbors=100)),
                               ('gbdt', GradientBoostingClassifier(),
                                ('kmc',
                                 KMeans(algorithm='DBSCAN', n_clusters=15)))],
                   final_estimator=XGBClassifier(base_score=None, booster=None,
                                                 callbacks=None,
                                                 colsample...
                                                 gpu_id=None, grow_policy=None,
                                                 importance_type=None,
                                                 interaction_constraints=None,
                                                 learning_rate=None,
                                                 max_bin=None,
                                                 max_cat_to_onehot=None,
                                                 max_delta_step=None,
                                                 max_depth=None,
                                                 max_leaves=None,
                                                 min_child_weight=None,
                                                 missing=nan,
                                                 monotone_constraints=None,
                                                 n_estimators=100, n_jobs=None,
                                                 num_parallel_tree=None,
                                                 predictor=None,
                                                 random_state=None,
                                                 reg_alpha=None,
                                                 reg_lambda=None, ...))
```

In [115…
```
y_pred3=clf.predict(X_test)
```

In [122…
```
y_pred3
```

Out[122]:
```
array([0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 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, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0], dtype=int64)
```

In [123…
```
accuracy_score(y_pred3,y_test)
```

Out[123]:
```
0.7653631284916201
```

In [118…
```
print(classification_report(y_pred,y_test))
print(confusion_matrix(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.92      0.74      0.82       136
           1       0.49      0.79      0.61        43

    accuracy                           0.75       179
   macro avg       0.71      0.77      0.71       179
weighted avg       0.82      0.75      0.77       179

[[101  35]
 [  9  34]]
```

In [135…
```
np.mean(cross_val_score(clf3, X,y, scoring='accuracy',cv=10 ))
```

Out[135]:
```
0.7184019975031212
```

Hyperimeter tunning with Genetic Algorithm Research

In [47]:
```
!pip install tpot
```

```
Collecting tpot
  Downloading TPOT-0.11.7-py3-none-any.whl (87 kB)
Requirement already satisfied: pandas>=0.24.2 in d:\anaconda1\lib\site-packages (from
tpot) (1.4.2)
Requirement already satisfied: scipy>=1.3.1 in d:\anaconda1\lib\site-packages (from t
pot) (1.7.3)
Collecting stopit>=1.1.1
  Downloading stopit-1.1.2.tar.gz (18 kB)
Collecting update-checker>=0.16
  Downloading update_checker-0.18.0-py3-none-any.whl (7.0 kB)
Requirement already satisfied: xgboost>=1.1.0 in d:\anaconda1\lib\site-packages (from
tpot) (1.6.2)
Collecting deap>=1.2
  Downloading deap-1.3.3-cp39-cp39-win_amd64.whl (114 kB)
Requirement already satisfied: numpy>=1.16.3 in d:\anaconda1\lib\site-packages (from
tpot) (1.21.5)
Requirement already satisfied: tqdm>=4.36.1 in d:\anaconda1\lib\site-packages (from t
pot) (4.64.0)
Requirement already satisfied: joblib>=0.13.2 in d:\anaconda1\lib\site-packages (from
tpot) (1.1.0)
Requirement already satisfied: scikit-learn>=0.22.0 in d:\anaconda1\lib\site-packages
(from tpot) (1.0.2)
Requirement already satisfied: pytz>=2020.1 in d:\anaconda1\lib\site-packages (from p
andas>=0.24.2->tpot) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in d:\anaconda1\lib\site-packag
es (from pandas>=0.24.2->tpot) (2.8.2)
Requirement already satisfied: six>=1.5 in d:\anaconda1\lib\site-packages (from pytho
n-dateutil>=2.8.1->pandas>=0.24.2->tpot) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda1\lib\site-packages
(from scikit-learn>=0.22.0->tpot) (2.2.0)
Requirement already satisfied: colorama in d:\anaconda1\lib\site-packages (from tqdm>
=4.36.1->tpot) (0.4.4)
Requirement already satisfied: requests>=2.3.0 in d:\anaconda1\lib\site-packages (fro
m update-checker>=0.16->tpot) (2.27.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in d:\anaconda1\lib\site-pac
kages (from requests>=2.3.0->update-checker>=0.16->tpot) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\anaconda1\lib\site-package
s (from requests>=2.3.0->update-checker>=0.16->tpot) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda1\lib\site-packages (from r
equests>=2.3.0->update-checker>=0.16->tpot) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda1\lib\site-packages
(from requests>=2.3.0->update-checker>=0.16->tpot) (2021.10.8)
Building wheels for collected packages: stopit
  Building wheel for stopit (setup.py): started
  Building wheel for stopit (setup.py): finished with status 'done'
  Created wheel for stopit: filename=stopit-1.1.2-py3-none-any.whl size=11956 sha256=
728558b5594529e0826b6297e591fb6cd684874f60ec73c4993869edb30fea26
  Stored in directory: c:\users\mrlaptop\appdata\local\pip\cache\wheels\48\8c\93\3afb
1916772591fe6bcc25cdf8b1c5bdc362f0ec8e2f0fd413
Successfully built stopit
Installing collected packages: update-checker, stopit, deap, tpot
Successfully installed deap-1.3.3 stopit-1.1.2 tpot-0.11.7 update-checker-0.18.0
```

In [90]:
```
from tpot import TPOTClassifier
```

In [111…
```
tpot = TPOTClassifier(generations=3, population_size=5,
verbosity=2, offspring_size=10,
scoring='accuracy', cv=5)
```

```
tpot.fit(X_train, y_train)
print(tpot.score(X_test, y_test))
```

```
Optimization Progress:   0%|          | 0/35 [00:00<?, ?pipeline/s]
Generation 1 - Current best internal CV score: 0.6952723333005023

Generation 2 - Current best internal CV score: 0.7079483896385306

Generation 3 - Current best internal CV score: 0.7079483896385306

Best pipeline: XGBClassifier(RobustScaler(input_matrix), learning_rate=0.01, max_dept
h=9, min_child_weight=13, n_estimators=100, n_jobs=1, subsample=0.9500000000000001, v
erbosity=0)
0.7318435754189944
```

Model Ensemble After Genetic Algorithm Search

In [131…
```
clf5 = StackingClassifier(
    estimators=estimators,
    final_estimator=xgb.XGBClassifier( learning_rate=0.01, max_depth=9, min_child_weig
    cv=10
)
```

In [132…
```
clf5.fit(X_train,y_train)
y_pred5=clf5.predict(X_test)
print(accuracy_score(y_pred5,y_test))

# As we can see the accuracy score has increased. It can increased with likes of Feart
```

```
0.7932960893854749
```

In [134…
```
print(classification_report(y_pred5,y_test))
print(confusion_matrix(y_pred5,y_test))
```

```
              precision    recall  f1-score   support

           0       0.92      0.78      0.85       129
           1       0.59      0.82      0.69        50

    accuracy                           0.79       179
   macro avg       0.76      0.80      0.77       179
weighted avg       0.83      0.79      0.80       179

[[101  28]
 [  9  41]]
```

In [133…
```
np.mean(cross_val_score(clf5, X,y, scoring='accuracy',cv=10 ))
```

Out[133]:
```
0.7026342072409488
```