Titanic Ensemble Stacking - Code + Research Paper

June 4, 2020

1 Please Note: The Research paper is at the end of the code and DA

2 Titanic Survival Prediction Using Ensemble Stacking

Name: Sidhantha Poddar Registration number: 17BCE2044

Abstract: We have several machine learning models available today which work in a very different way and produce a different result. Considering a given data set some models might work for one case while other work and might not work for all the cases. If we can stack different models with their trained parameters, then we might be able to achieve a model with better validation accuracy. This paper presents an approach that uses multiple classification algorithms and combines them, in order to impart the final model with better accuracy and efficiency, avoiding overfitting. Here, we will be using the Titanic dataset for our project which consists of 12 columns; 11 attributes and 1 result column. We will further use the concept of feature engineering in which we will try to derive new attributes out of the available ones. After using multiple models for making the predictions we will make use a different approach to the ensemble to tune our final predictive model.

COMPARATIVE STUDY /RESULTS AND DISCUSSION We yielded a good score in terms of accuracy, 84% which clears us off the over-fitting trap. Upon using the Max Voting Ensemble Technique, we got a very high accuracy of 98.89% which was highly inclined to becoming a case of overfitting which is why we did not follow that path. Also, Precision itself cannot be one true metric to look at the performance of a model since accuracy takes in the assumption that the attributes are fairly distributed over the dataset which is hardly ever the case. Precision and Reminiscence should also be considered. Upon dropping the XGBoost in the Average Voting case, the accuracy also fell to a mere 40% which is also clearly an instance of under-fitting; a poor performing model. Hence our model worked better in comparison with Simple Ensemble Techniques.

3 Coding

3.1 Importing Libraries

[9]:
```
import pandas as pd
import numpy as np
import statistics
from sklearn import preprocessing
```

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
 ↪GradientBoostingClassifier, ExtraTreesClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('bmh')
%matplotlib inline
```

[42]:
```
train = pd.read_csv('input/train.csv')
test = pd.read_csv('input/test.csv')
test_id = test['PassengerId']
combine = [train, test] # combine train and test data, easy to do data␣
 ↪manipulation
```

[43]:
```
train.head()
```

[43]:

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

[44]:
```
titanic_df=train
titanic_df.info()
```

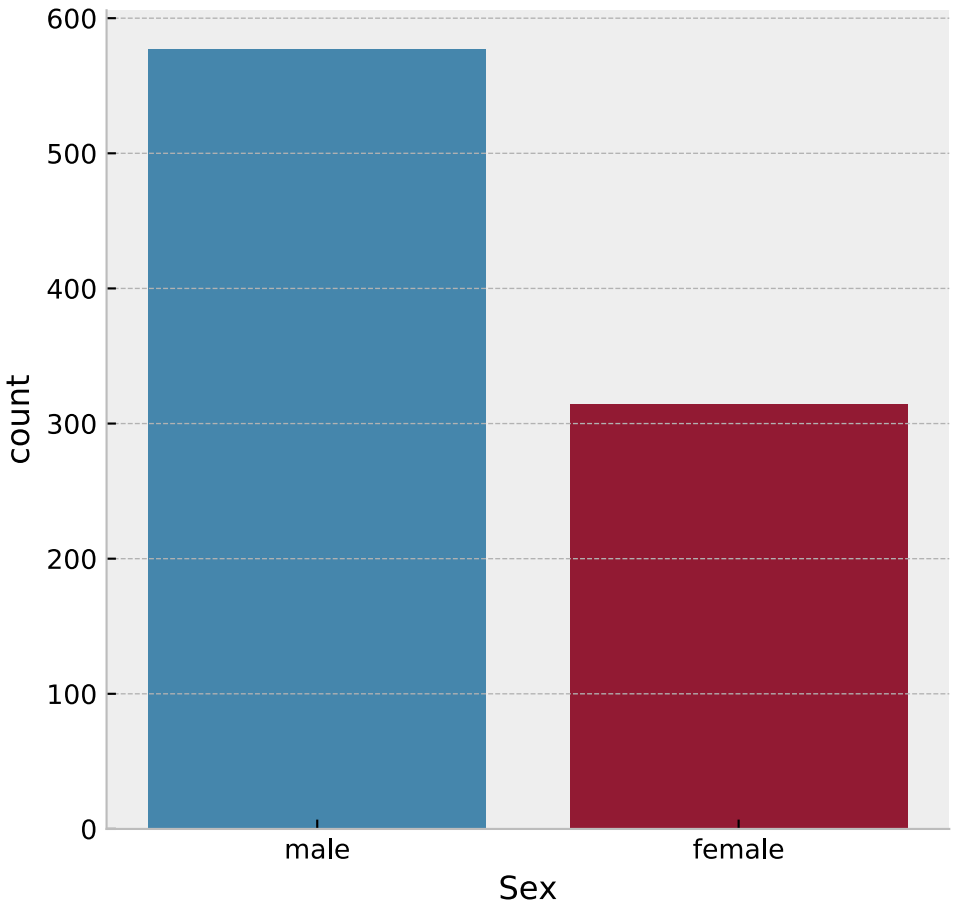
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
```

```
---   ------       --------------  -----
 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
```

3.2 Dataset vizualizations and analysis

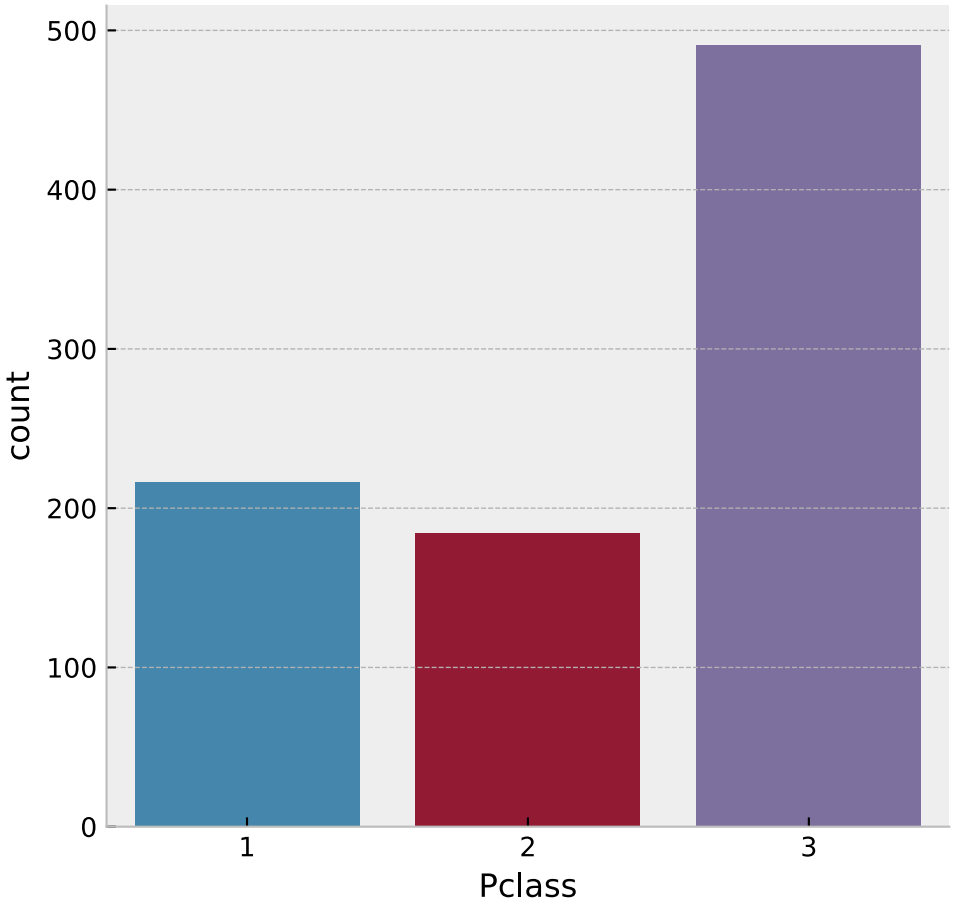
[45]:
```
sns.factorplot('Sex',data=titanic_df,kind='count')
```

[45]:
```
<seaborn.axisgrid.FacetGrid at 0x7f56757d6210>
```



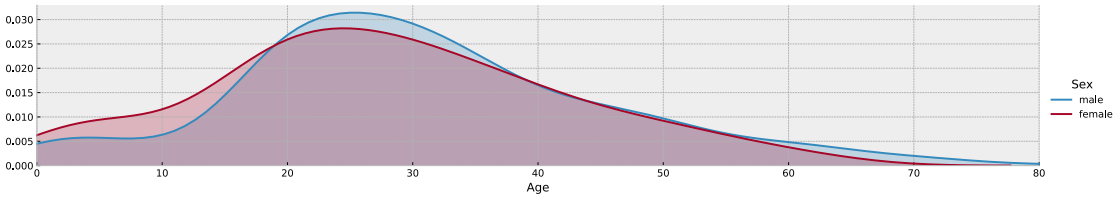
[46]:
```
sns.factorplot('Pclass',data=titanic_df,kind='count')
```

[46]: <seaborn.axisgrid.FacetGrid at 0x7f567577ef90>



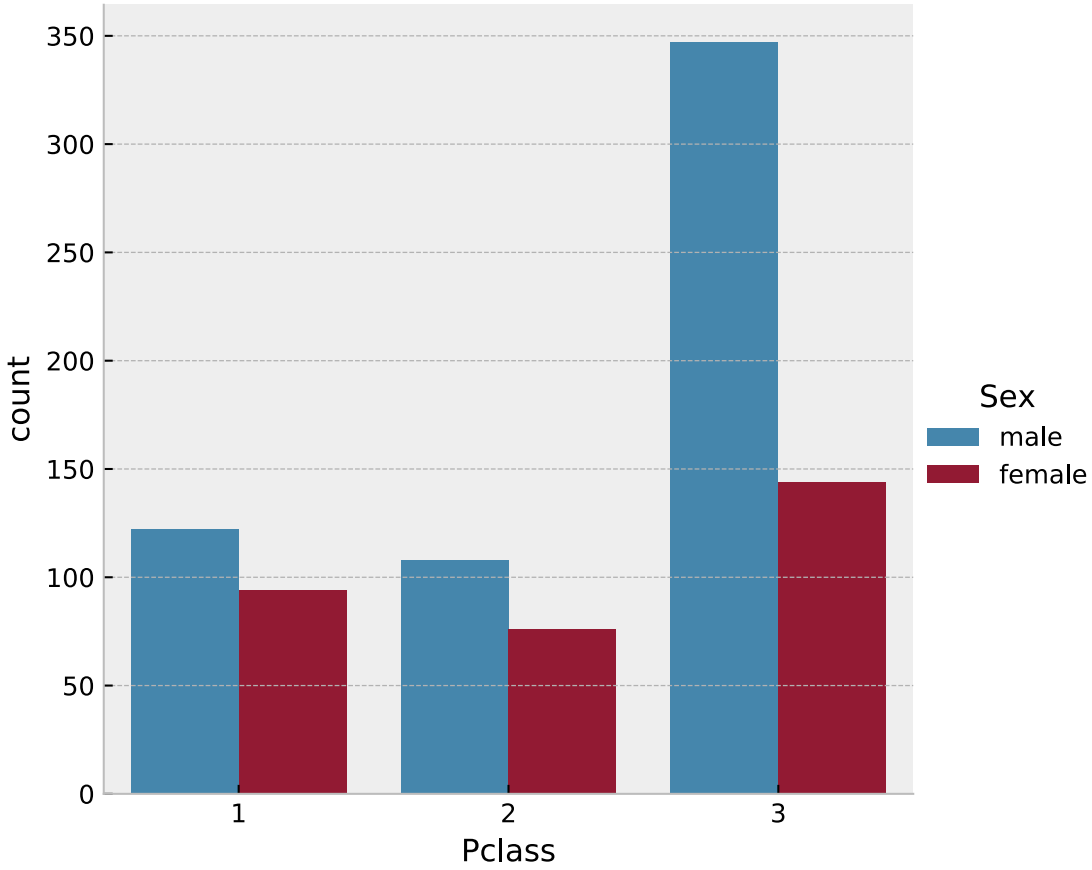
[47]:
```
as_fig = sns.FacetGrid(titanic_df,hue='Sex',aspect=5)

as_fig.map(sns.kdeplot,'Age',shade=True)

oldest = titanic_df['Age'].max()

as_fig.set(xlim=(0,oldest))

as_fig.add_legend()
```

[47]: <seaborn.axisgrid.FacetGrid at 0x7f566be28750>



[48]:
```
sns.factorplot('Pclass',data=titanic_df,hue='Sex',kind='count')
```

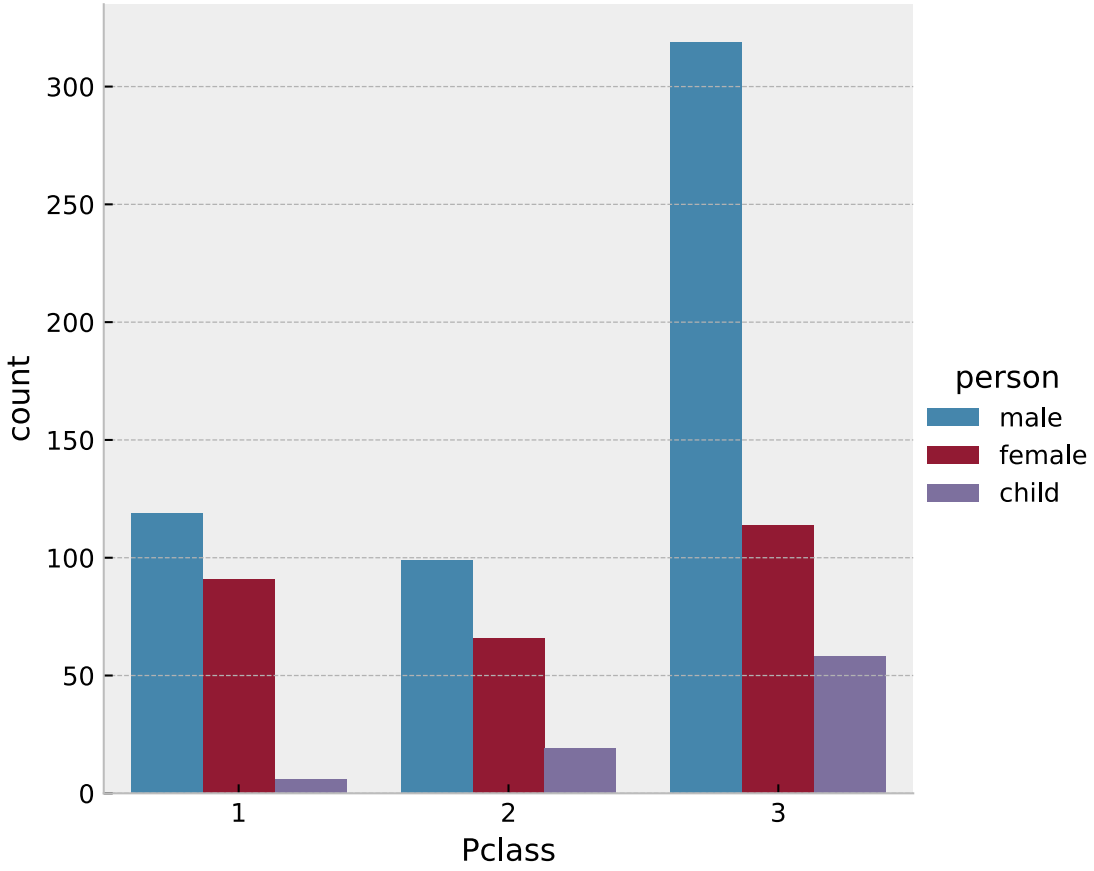
[48]:
```
<seaborn.axisgrid.FacetGrid at 0x7f566c34b610>
```



[49]:
```
def titanic_children(passenger):

    age , sex = passenger
    if age <16:
        return 'child'
    else:
```

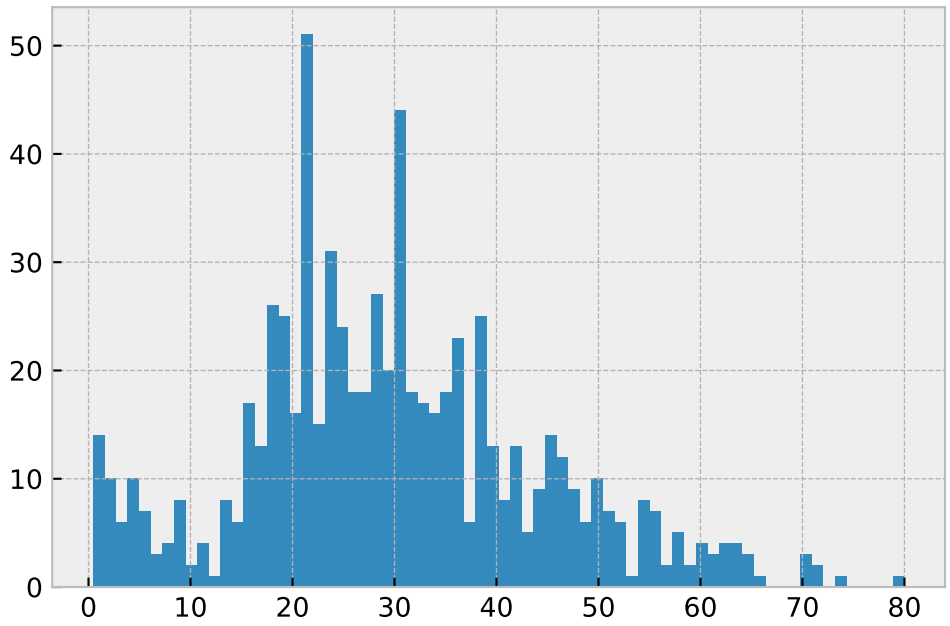
```
        return sex

titanic_df['person'] = titanic_df[['Age','Sex']].apply(titanic_children,axis=1)
sns.factorplot('Pclass',data=titanic_df,hue='person',kind='count')
```

[49]: <seaborn.axisgrid.FacetGrid at 0x7f566c34be10>



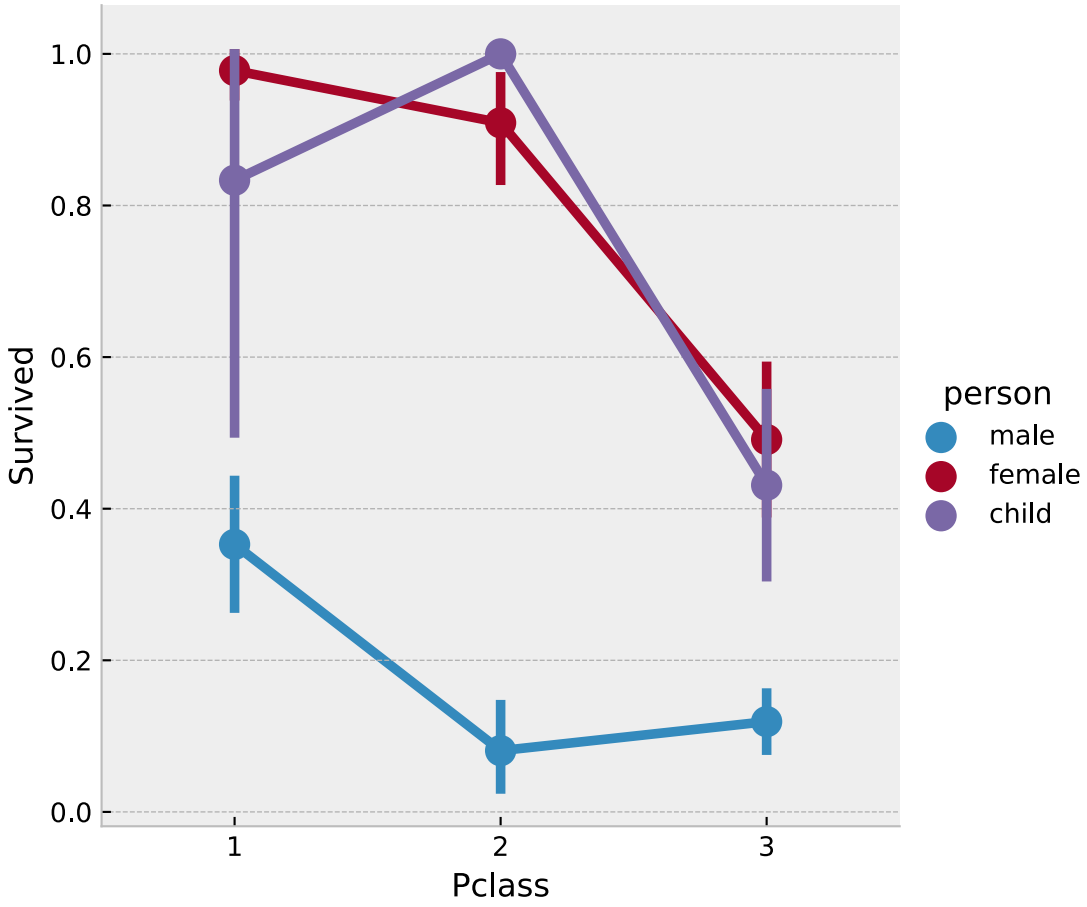
[50]:
```
titanic_df['Age'].hist(bins=70)
```

[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7f566c190090>



[51]:
```
sns.factorplot('Pclass','Survived',data=titanic_df,hue='person')
```

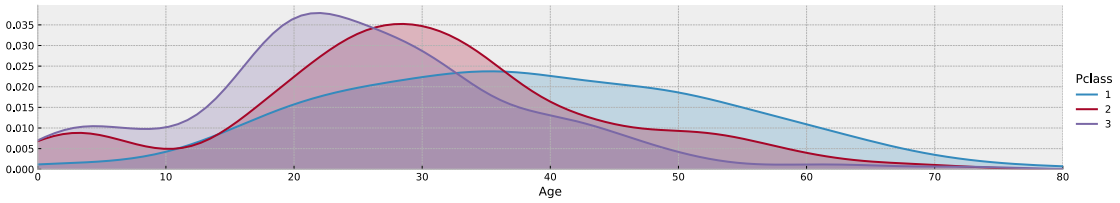
[51]:
```
<seaborn.axisgrid.FacetGrid at 0x7f566c50dc10>
```



The above graph shows that the survival rate for male is very low nevertheless of the class. And, the survival rate is less for the 3rd class passengers.

[52]:
```
as_fig = sns.FacetGrid(titanic_df,hue='Pclass',aspect=5)

as_fig.map(sns.kdeplot,'Age',shade=True)

oldest = titanic_df['Age'].max()

as_fig.set(xlim=(0,oldest))

as_fig.add_legend()
```

[52]:
```
<seaborn.axisgrid.FacetGrid at 0x7f566bda2c50>
```



From the above graphs, we can say that there are more number of passengers with a age group of 20 to 40 in all of the three classes which we are predicting.

3.3 Data Augumentation and preprocessing

Combining sibsp and parch to familysize to get better estimate

[53]:
```
for df in combine: # add feature 'FamilySize'
    df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
trans1=combine[0]
trans1.head()
```

[53]:

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

adding new feature alone which tells us that if he travalled alone or not

[54]:
```
for df in combine: # add feature 'Alone'
    df['Alone'] = 0
    df.loc[df['FamilySize'] == 1, 'Alone'] = 1
trans2=combine[0]
```

```
trans2.head()
```

[54]:

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

[55]:
```
for df in combine: # fill missing values for 'Embarked'
    df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
trans3=combine[0]
print(trans3.info())
trans3.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 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     891 non-null    object
 12  person       891 non-null    object
```

```
 13  FamilySize   891 non-null    int64
 14  Alone        891 non-null    int64
dtypes: float64(2), int64(7), object(6)
memory usage: 104.5+ KB
None
```

[55]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked  person  FamilySize  Alone
0      0         A/5 21171   7.2500   NaN        S    male           2      0
1      0          PC 17599  71.2833   C85        C  female           2      0
2      0  STON/O2. 3101282   7.9250   NaN        S  female           1      1
3      0            113803  53.1000  C123        S  female           2      0
4      0            373450   8.0500   NaN        S    male           1      1
```

[56]:
```
for df in combine: # fill missing values for 'Fare' and transform into␣
 ↪categorical feature
    df['Fare'].fillna(df['Fare'].median(), inplace=True)
    df['farecat'] = 0
    df.loc[df['Fare'] <= 10.5, 'farecat'] = 0
    df.loc[(df['Fare'] > 10.5) & (df['Fare'] <= 21.679), 'farecat'] = 1
    df.loc[(df['Fare'] > 21.679) & (df['Fare'] <= 39.688), 'farecat'] = 2
    df.loc[(df['Fare'] > 39.688) & (df['Fare'] <= 512.329), 'farecat'] = 3
    df.loc[df['Fare'] > 512.329, 'farecat'] = 4

trans4=combine[0]
trans4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 16 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     891 non-null    object
 12  person       891 non-null    object
 13  FamilySize   891 non-null    int64
 14  Alone        891 non-null    int64
 15  farecat      891 non-null    int64
dtypes: float64(2), int64(8), object(6)
memory usage: 111.5+ KB
```

[57]:
```
for df in combine: # fill missing values for 'Age' and transform into␣
 ↪categorical feature
    avg = df['Age'].mean()
    std = df['Age'].std()
    NaN_count = df['Age'].isnull().sum()

    age_fill = np.random.randint(avg-std, avg+std, NaN_count)
    df.loc[df['Age'].isnull(), 'Age'] = age_fill
    df['Age'] = df['Age'].astype(int)

    df['agecat']=0
    df.loc[df['Age'] <= 16, 'agecat'] = 0
    df.loc[(df['Age'] > 16) & (df['Age'] <= 32), 'agecat'] = 1
    df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'agecat'] = 2
    df.loc[(df['Age'] > 48) & (df['Age'] <= 64), 'agecat'] = 3
    df.loc[df['Age'] > 64, 'agecat'] = 4
trans5=combine[0]
trans5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 17 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          891 non-null    int64
 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     891 non-null    object
 12  person       891 non-null    object
 13  FamilySize   891 non-null    int64
 14  Alone        891 non-null    int64
 15  farecat      891 non-null    int64
 16  agecat       891 non-null    int64
dtypes: float64(1), int64(10), object(6)
memory usage: 118.5+ KB
```

[58]:
```
import re
def only_title(name): # manipulation 'Name', extracting titles from names
    title = re.findall(' ([A-Za-z]+)\.', name)
    if title:
        return title[0]

for df in combine:
    df['Title'] = df['Name'].apply(only_title)

for df in combine:
    df['Title'] = df['Title'].replace(['Lady', 'Countess','Capt', 'Col', 'Don',␣
 ↪'Dr', 'Major',
                                       'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    df['Title'] = df['Title'].replace('Mlle', 'Miss')
    df['Title'] = df['Title'].replace('Ms', 'Miss')
    df['Title'] = df['Title'].replace('Mme', 'Mrs')

############ Encoding features, make them ready for classifiers
# feature_drop = ['PassengerId', 'Name', 'SibSp', 'Parch', 'Ticket', 'Cabin',␣
 ↪'FamilySize']
# for df in combine:
#     df.drop(feature_drop, axis=1, inplace=True)

trans6=combine[0]
trans6.head()
```

[58]:

	PassengerId	Survived	Pclass	\
0	1	0	3	
1	2	1	1	
2	3	1	3	
3	4	1	1	
4	5	0	3	

	Name	Sex	Age	SibSp	\
0	Braund, Mr. Owen Harris	male	22	1	

```
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female   38      1
2                             Heikkinen, Miss. Laina  female   26      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female   35      1
4                           Allen, Mr. William Henry    male   35      0

   Parch            Ticket     Fare Cabin Embarked  person  FamilySize  Alone  \
0      0         A/5 21171   7.2500   NaN        S    male           2      0
1      0          PC 17599  71.2833   C85        C  female           2      0
2      0  STON/O2. 3101282   7.9250   NaN        S  female           1      1
3      0            113803  53.1000  C123        S  female           2      0
4      0            373450   8.0500   NaN        S    male           1      1

   farecat  agecat Title
0        0       1    Mr
1        3       2   Mrs
2        0       1  Miss
3        3       2   Mrs
4        0       2    Mr
```

[59]:
```
for df in combine: # add feature 'Alone'
    df['gender'] = 0
    df.loc[df['Sex'] == "male", 'gender'] = 1
trans6=combine[0]
```

[60]:
```
trans6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 19 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          891 non-null    int64
 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     891 non-null    object
 12  person       891 non-null    object
 13  FamilySize   891 non-null    int64
 14  Alone        891 non-null    int64
 15  farecat      891 non-null    int64
```

```
 16  agecat       891 non-null    int64
 17  Title        891 non-null    object
 18  gender       891 non-null    int64
dtypes: float64(1), int64(11), object(7)
memory usage: 132.4+ KB
```

3.4 Ensemble stacking method

[61]:
```
def accuracy(k,prediction):
    match=0
    i=0
    for x in prediction:
        if x==k[i]:
            match+=1
        i+=1
    return(match/k.size)
```

Creating multiple models for stacking

[62]:
```
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
X  = train[['gender','Pclass','agecat']]
X1 = test[['gender','Pclass','agecat']]
Y = train[['Survived']]
clf1 = tree.DecisionTreeClassifier()
clf1 = clf1.fit(X,Y)
pred1 = clf1.predict(X)
tpred1 = clf1.predict(X1)
k=np.array(Y)
print(accuracy(k,pred1))
```

```
0.8013468013468014
```

[63]:
```
X  = train[['gender','Pclass','farecat',]]
X2 = test[['gender','Pclass','farecat',]]
Y = train[['Survived']]
X.insert(1, 'pred1', pd.DataFrame({"pread1":pred1}))
X2.insert(1, 'pred1', pd.DataFrame({"pread1":tpred1}))
clf2 = tree.DecisionTreeClassifier()
clf2 = clf2.fit(X,Y)
pred2 = clf2.predict(X)
tpred2 = clf2.predict(X2)
print(accuracy(k,pred2))
```

```
0.8215488215488216
```

[64]:
```
X  = train[['gender','Pclass','farecat','FamilySize']]
X3 = test[['gender','Pclass','farecat','FamilySize']]
```

```
Y = train[['Survived']]
clf3=RandomForestClassifier(n_estimators=100)
clf3.fit(X,Y)
pred3=clf3.predict(X)
tpred3 = clf3.predict(X3)
print(accuracy(k,pred3))
```

0.8305274971941639

[65]:
```
X  = train[['gender','FamilySize','agecat','Pclass']]
Y = train[['Survived']]
X4 = test[['gender','FamilySize','agecat','Pclass']]
clf4=RandomForestClassifier(n_estimators=1000)
clf4.fit(X,Y)
pred4=clf4.predict(X)
tpred4=clf4.predict(X4)
print(accuracy(k,pred4))
```

0.8361391694725028

[66]:
```
from sklearn import linear_model
X  = train[['gender','FamilySize','agecat','Pclass','Parch','SibSp','Fare']]
Y = train[['Survived']]
X5=test[['gender','FamilySize','agecat','Pclass','Parch','SibSp','Fare']]
reg = linear_model.LinearRegression()
reg.fit(X,Y)
pred5=reg.predict(X)
tpred5=reg.predict(X5)
print(accuracy(k,pred5))
```

0.0

[67]:
```
from sklearn.linear_model import LogisticRegression
X  = train[['gender','FamilySize','agecat','Pclass','Parch','SibSp','Fare']]
Y = train[['Survived']]
X6=test[['gender','FamilySize','agecat','Pclass','Parch','SibSp','Fare']]
lreg = LogisticRegression(random_state=0,␣
 ↪solver='lbfgs',multi_class='multinomial')
lreg.fit(X,Y)
pred6=lreg.predict(X)
tpred6=lreg.predict(X5)
print(accuracy(k,pred6))
```

0.7890011223344556

[68]:
```
arr=[]
for i in pred5:
    arr=arr+ [i]
```

```
print(accuracy(pred1,pred2))
print(accuracy(pred1,pred3))
print(accuracy(pred1,pred4))
print(accuracy(pred1,pred6))
print(accuracy(pred2,pred3))
print(accuracy(pred2,pred4))
print(accuracy(pred2,pred6))
print(accuracy(pred3,pred4))
print(accuracy(pred3,pred6))
print(accuracy(pred4,pred6))
```

```
0.9169472502805837
0.867564534231201
0.8888888888888888
0.8462401795735129
0.9281705948372615
0.9494949494949495
0.9113355780022446
0.9405162738496072
0.920314253647587
0.8900112233445566
```

[69]:
```
totalcalc=[]
for i in range(0,len(pred1)):
    a1=pred1[i]
    a1+=pred2[i]*2**1
    a1+=pred3[i]*2**2
    a1+=pred4[i]*2**3
    a1+=pred6[i]*2**4
    totalcalc+=[a1]
```

[71]:
```
np.array(set(totalcalc))
```

[71]:
```
array({0, 1, 3, 4, 8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 26,
27, 28, 29, 30, 31},
      dtype=object)
```

[73]:
```
arrx0=[0 for i in range(0,32)]
arrx1=[0 for i in range(0,32)]
for i in range(0,len(totalcalc)):
    if k[i]==1:
        arrx1[totalcalc[i]]+=1
    else:
        arrx0[totalcalc[i]]+=1

arrxf=[0 for i in range(0,32)]
for i in range(0,32):
```

```
    if arrx1[i]>arrx0[i]:
        arrxf[i]=1
    else:
        arrxf[i]=0
```

[74]:
```
for i in range(0,32):
    print(i,"\t",arrx1[i],"\t",arrx0[i])
```

```
0        71      450
1        1       8
2        0       0
3        0       1
4        5       2
5        0       0
6        0       0
7        0       0
8        8       3
9        0       0
10       0       1
11       6       0
12       9       1
13       0       0
14       1       0
15       4       0
16       10      21
17       0       4
18       2       4
19       0       2
20       9       10
21       0       0
22       2       0
23       0       1
24       1       2
25       0       0
26       0       1
27       1       1
28       2       5
29       1       1
30       28      20
31       181     11
```

[75]:
```
trtotalcalc=[]
for i in range(0,len(pred1)):
    a1=pred1[i]
    a1+=pred2[i]*2**1
    a1+=pred3[i]*2**2
    a1+=pred4[i]*2**3
```

```
    a1+=pred6[i]*2**4
    trtotalcalc+=[arrxf[a1]]
accuracy(k,trtotalcalc)
```

[75]: 0.8484848484848485

[76]:
```
ttotalcalc=[]
for i in range(0,len(tpred1)):
    a1=tpred1[i]
    a1+=tpred2[i]*2**1
    a1+=tpred3[i]*2**2
    a1+=tpred4[i]*2**3
    a1+=tpred6[i]*2**4
    ttotalcalc+=[arrxf[a1]]
```

3.4.1 Results after Stacking

[77]:
```
print(accuracy(pred1,trtotalcalc))
print(accuracy(pred2,trtotalcalc))
print(accuracy(pred3,trtotalcalc))
print(accuracy(pred4,trtotalcalc))
print(accuracy(pred6,trtotalcalc))
```

```
0.8877665544332211
0.9528619528619529
0.9483726150392817
0.9719416386083053
0.8686868686868687
```

[78]:
```
dummy_data1 = {
        'pred1': pred1,
        'pred2': pred2,
        'pred3':pred3,
        'pred4':pred4,
        #'pred5':arr
        }

df1 = pd.DataFrame(dummy_data1)
clf = tree.DecisionTreeClassifier()
clf = clf.fit(df1,Y)
```

predicting on test data

[79]:
```
predt=clf.predict(df1)
print(accuracy(k,predt))
```

```
0.8428731762065096
```

3.4.2 Results

the accuraccy increased from 78% to 84% after stacking models

[ ]:

Titanic Survival Prediction Using Ensemble Stacking

Abstract

We have several machine learning models available today which work in a very different way and produce a different result. Considering a given data set some models might work for one case while other work and might not work for all the cases. If we can stack different models with their trained parameters, then we might be able to achieve a model with better validation accuracy.

This paper presents an approach that uses multiple classification algorithms and combines them, in order to impart the final model with better accuracy and efficiency, avoiding overfitting. Here, we will be using the Titanic dataset for our project which consists of 12 columns; 11 attributes and 1 result column. We will further use the concept of feature engineering in which we will try to derive new attributes out of the available ones. After using multiple models for making the predictions we will make use a different approach to the ensemble to tune our final predictive model.

I. INTRODUCTION

The most infamous event in the history of waterways was the wreckage of the ship Titanic. Boasting of being one of the worlds grandest and most powerful ships the titanic was subjected to admiration from all over the world. This resulted in great attention from media from every corner of the world. However due to unforeseen circumstances the Titanic met with a disastrous end, thus ending the lives of thousands of people on board.

In this project we are first analyzing the dataset, getting the most important features which are responsible for influencing the prediction (survival/not survival). For dataset analysis, we will be visualizing the data. Data visualization is basically done in order to understand the data. After analyzing the dataset, we combine two or more attributes to form new attributes and check its relationship with the prediction. So, we get a dataset containing new relevant attributes. We further perform analysis on the new dataset and get the relation of the newly combined attributes and the prediction of survival. Upon getting the best combination we use different machine learning models on each combination depending upon the attributes in the combination. Upon the selection of the combinations and their respective models we train and predict results on both training and testing dataset. We use the new column of predicted data as another attribute for a new model. Upon multiple predictions and nested predistortions, we select the best set of predictions and used the max voting feature and train a model based on the selected predictions sets to get the result

By weighing the contributions of each sub-model to the combined prognosis by the expected performance of the sub-model. This can be extended even further by training an entirely new model to learn how to best combine the contributions from each sub-model.

A curb of this approach is that each model contributes nearly the same amount to the ensemble prediction, regardless of how well the model performs. This approach, with some variations, is called a weighted average ensemble, which weighs the contribution of each ensemble member by the trust or expected performance of the model on a given dataset. This allows better performing models to contribute more and less-better-performing models to contribute less. The weighted average ensemble is an improvement over the model average ensemble.

Often the level 2 model will perform better than single models due to its smoothing/averaging nature and the facility to show each base model where it performs the best and discredit each base model where it performs in a bad way. Due to this reason, stacking of multiple algorithms is the most effective

and efficient method when the base models are largely different.

II. BACKGROUND

Logistic Regression or we can say multivariate analysis is a regression analysis conducted when the output is a binary variable. Like all regression analyses, the logistical regression may be a prognostication analysis. logistic regression is employed to explain data and to elucidate the link between one dependent binary variable and one or a lot of nominal, ordinal, interval or ratio-level non-dependent variables.

A decision tree is to boot a flowchart-like structure within that each internal node represents a "test" on associate attribute (e.g. whether or not or not or not or not a coin flip comes up heads or tails), each branch represents the results of the check,and each leaf node represents a category label (decision was taken once computing all attributes). The route from root to leaf represent organization rules.

Random forests is one of ensemble stacking method which combines multiple decision to create a singular tree. And other different tasks that operate by constructing many decision trees during training time and outputting the category that's the mode of the categories (classification) or means prediction (regression) of the individual trees.

Literature Survey

Many techniques have been proposed for using multiple machine learning models. These techniques come under the category of Ensemble Learning. One of the most uncomplicated Ensemble methods are Max Voting, Averaging, and Weighted Averaging.

In machine learning, there are several algorithms – models which could be used to predict values based on the data they receive. Some are Strong leaners of data and some are weak learners.

Strong learners such as Neural Network have a great prediction accuracy but they are very slow as compared to other algorithms

Weak learners are algorithms which process data very fast as compared to the Strong learners but have a less prediction accuracy. eg decision tree, random forest, SVM, gradient boost, KNN, K Means, Regression

Ensemble Stacking algorithm combines various Weak learner algorithms to get a Strong prediction value.

Algorithms can be stacked in various ways. Such as

1. Prediction results of one algorithm could be used for another algorithm, Eg Clusters formed by K-means could be used by Decision tree to form classification.
2. Statistical Algorithms such as KNN, regression could be used to form new attributes/fields which could intern be used for calculation by algorithms. Eg if we have age as a factor, K Means could be used to classify age into age-groups which could prevent overfitting OR regression could be used to combine age and weight (BMI) to form a new field.
3. We can also have many algorithms to predict their independent results and then combine them based on the maximum count or their average

Ensemble voting algorithm is generally used in cases of classification where each algorithm cast is its vote ie its classification {0,1} based on all the votes the net results/prediction is calculated.

There and more than one way in which stacking of votes can be done

Vote counting -Votes could be counted for each algorithm and the classification with maximum votes would be selected.

The max voting method is normally used in classification-questions(or problems). This method, in which, more than one models are used to give results(predictions) for each data point. The results(predictions) given by each model is regarded as a ‘vote’. The results(predictions) counted from most of the models are considered to be the final prediction.

Almost same as to the max voting method, a bunch of predictions is done for each data point in the technique of averaging. In this method, we will take a mean of the predictions from all the chosen models and will use it to give our final prognosis. The averaging method can be done for making prognosis in regression. Or during, the calculation of probabilities for classification questions.

This is an improvement of well known averaging technique. All the models are given with dissimiliar weights telling the significance of each model for prediction.

Other Advanced Techniques of Ensemble Learning are Stacking, Blending and Bagging. Stacking is nothing but an ensemble learning method which

uses results(predictions) from more than one model (Eg: decision tree method and K-NN method) to create a completely new model. This model is to be used for deriving results(predictions) on given testing dataset.

Blending differs from stacking such that in this approach it makes use of only the validation data obtained from the training set. This ensures that the predictions do not suffer a biased nature and makes our analysis strong. The predictions are calculated based on the holdout dataset only. The holdout dataset and the predictions are used to create new model which we run on the testing dataset.

Bagging is a method used to combine multiple algorithms to predict a combined single genrelized result. Bootstrapping is a sampling method, here we create subsets of observations from the original dataset, with replacement. The size of the subsets is the same as the size of the original set.

The following table explains all the above techniques in detail and how they calculate their output.

Name of technique	**Explanation/Output**
Max Voting	The most occurring output is chosen as the output
Average Voting	Average of all outputs is chosen as the final output
Weighted Average Voting	Believed to be Stronger Models are given more weight which in result are given more importance when computing the average.
Stacking	The output of many models is used as features themselves and fed to another model to give outputs
Blended	Like Stacking but uses only a holdout set from the train set to make predictions
Bagging	Bagging uses a combination of the results of all models by Bootstrapping

IV. DATASET DESCRIPTION & SAMPLE DATA

The most infamous event in the history of waterways was the wreckage of the ship Titanic. Boasting of being one of the worlds grandest and most powerful ships the titanic was subjected to admiration from all over the world. This resulted in great attention from media from every corner of the world. However due to unforeseen circumstances the Titanic met with a disastrous end, thus ending the lives of thousands of people on board.
We have been given with 2 datasets; Training Dataset and Testing Dataset. Training Dataset contains an extra output column telling whether the given-on board passenger survived or not. The Testing dataset contains data on 418 onboard passengers whereas the Training Set contains data on 891 onboard passengers

- (Numerical data) **PassengerId** - unique Id for every passenger
- (Numerical data) **Survived** - Survived or Not
- (Numerical data) **Pclass** - Class of Travel
- (String type) **Name**- Name of Passenger
- (String type) **Sex** - Gender
- (Numerical data) **Age** - Age of Passengers
- (String data) **SibSp** - Number of Sibling/Spouse aboard
- (Numerical data) **Parch** - Number of Parent/Child aboard
- (String type) **Ticket** - Ticket No.
- (Numerical data) **Fare** - Price Paid For Ticket
- (String type) **Cabin** - Cabin Assigned

V. PROPOSED ALGORITHM WITH FLOWCHART

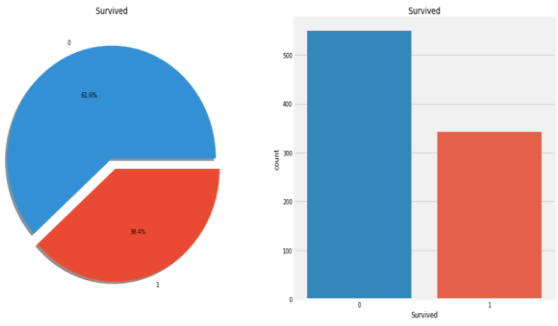
We will begin with our Data Acquisition which involves getting our Training and Test Sets from

Kaggle. Then we move on with the beginning of our Data Preprocessing which involves Data Cleaning and Data Manipulation. Data Cleaning deals with Quality issues in the dimensions of Completeness, Validation, Accuracy, and Consistency.

Data Manipulation deals with the issues of Tidiness; The Structure/Organisation of the table itself. We will deal with the Missing and Invalid data first and then move further to structure the table. After completing these steps, we are complete with our Data Preprocessing and can move on to our Next Step;

EDA (Exploratory Data Analysis). This involves finding relations between variables and trying to understand how each variable affects the outcome and to what magnitude. We will try to find the effect on the quantitative variable due to differing qualitative variables. This is a very important step. This will help us to gain insight into which features hold the most important and allow us to build upon new features via Feature Engineering. We add new features such as

1. **Family member** adding siblings and parch.
2. **Alone** by counting family members
3. **AgeCat, FareCat** by categorizing age and fare paid into groups this is done to prevent overfitting or data learning



After Gaining the insights and successfully converting them into useful and important new features we can move on to the prime part of our report. Model Building.

We use 5 different models with different attributes and predicts survival or not. We also use the concept of nested stacking which is using the predicted result of one model as an attribute of another. This way we create 5 models and predict the results for both training and testing dataset for each model

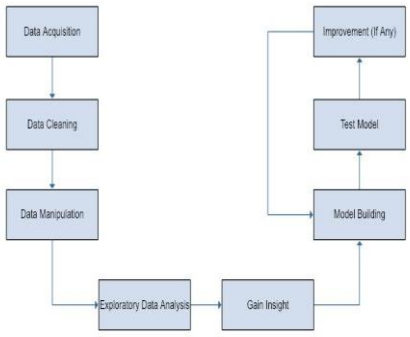
Now having 5 independent results for each model we create a new ML model using these 5 predicted results and having a weighted average of each cluster that falls in their respective algorithm the result would be calculated

After getting the results via Kaggle, we can decide if our model is weak or potentially overfitting. If it's not up to the mark, which is subjective, in this case, a good score would be between (75% - 85%), we will further reiterate to get a better score.

VI. EXPERIMENTS RESULTS

Data Cleaning¶

1. When real world problem data we receive contains more errors than we expect.
2. We endlessly need to predict right values, assign missing ones, finding links between various data artifacts such as schematics and records.
3. We need to change the habit of treating data cleaning as a fragmentary workout (solving diverse types of errors in separation), and instead leverage all signals and resources.
4. The main purpose of data cleaning is to detect and eliminate errors and irregularities to increase the value of data



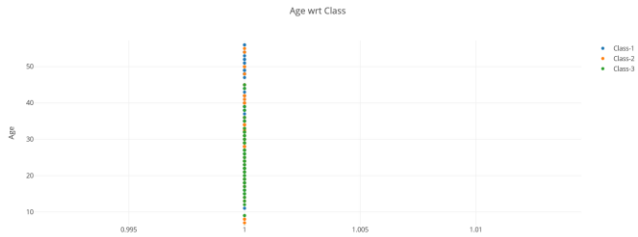
in analytics and decision making.

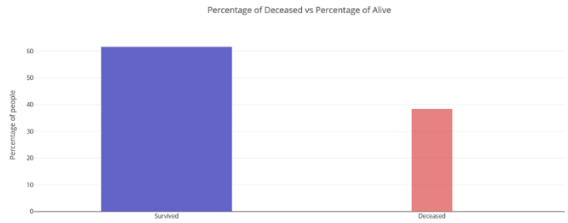
Since the data acquired itself is pretty much clean, we won't have to do much besides imputing the missing values in the ages. There are missing values in the Cabin Column as well, but that is because some people were sleeping in berths due to lower class tickets hence, it is fine

Exploratory Data Analysis

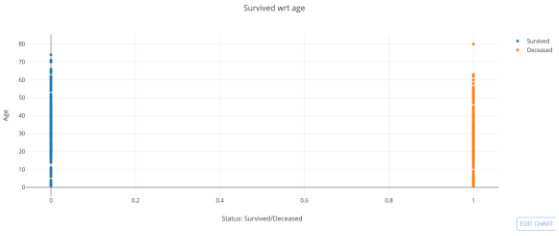
After cleaning the Dataset properly, we can analyze our data easily without any problems. Let's begin by visualizing the relative and absolute frequencies of our outcome.

We can see that most of our passengers were unable to survive the calamity, unfortunately. Let's take deep dive into our dataset to further understand what factors were contributing to the survival chance of a passenger.

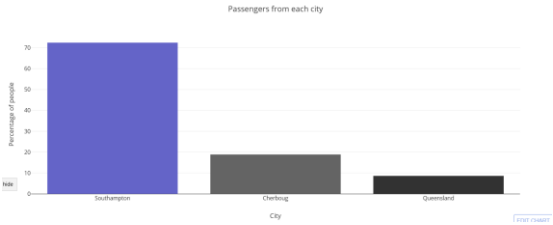




We can easily infer that most of the people on board died that is nearly 62%



People who were old had more chances of dying

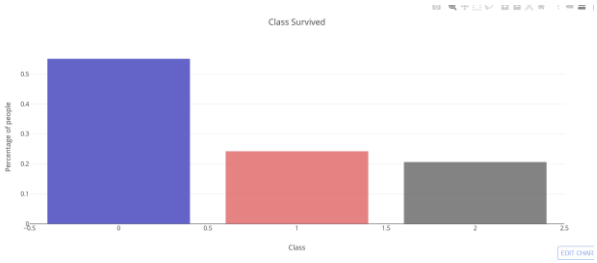


Titanic stopped at 3 places

70% of people boarded the giant at Southampton

20% of people boarded at Cherbourg

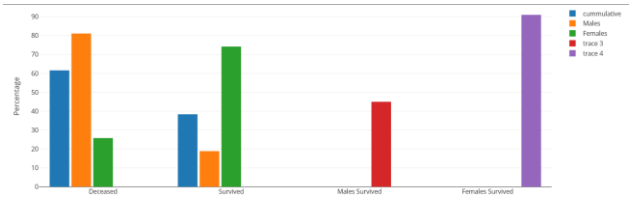
And the rest 10% boarded at Queensland



More than 50% of the people from class 1 survived

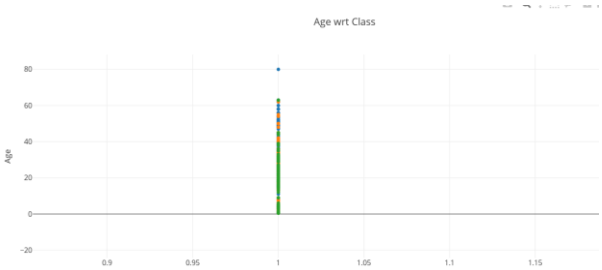
And more than 20% of the people from class 3 survived

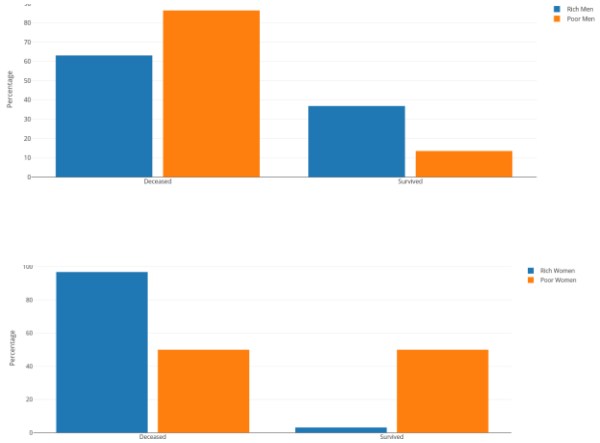
And only 25% of the people from class 2 survived



Trace 3 and Trace 4 show cumulative survival

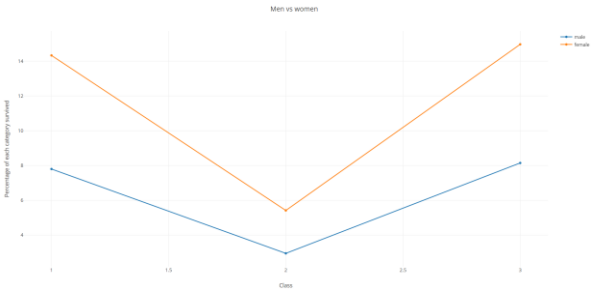
Gender played a huge role in surviving the downfall, we can easily see that from the above chart. If you're a female then you're more likely to survive more than a male



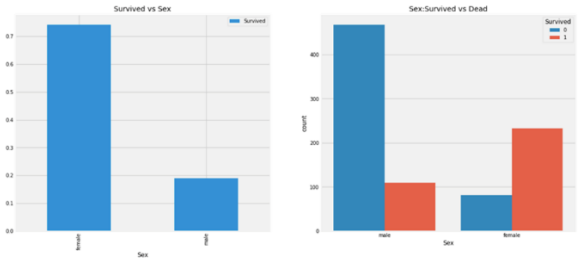


Money can play a huge role in saving a life. We can easily infer from the above graphs that rich people tend to survive more. That's the case in men

Exactly the opposite scenario follows for the women

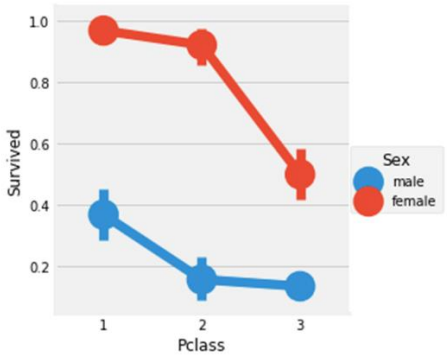


We can easily infer that more percentage of people survived from class 3 and class 1 when compared to class 2



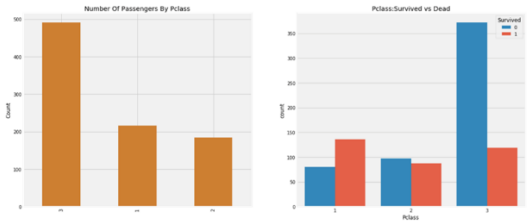
First, we must look at the gender and see if we can find out a hidden factor.

The number of men on the ship is a lot more than the number of women. Still, the number of women saved is around 2 times the number of men saved. The survival rates for women on the ship is around 75% while that for men is around 18-19%.



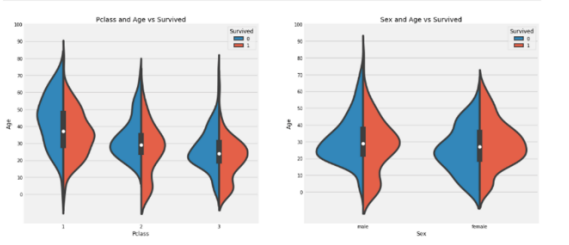
People say Money Can't Buy Everything. But we can clearly see that Passengers of Pclass 1 was given a very high priority while rescue. Even though the number of Passengers in Pclass 3 were a lot higher, still the number of survivals from them is very low, somewhere around **25%.**

For Pclass-1 percentage survived is around **63%** while for Pclass2 its around **48%**. So, money and status matter.



Observing at the CrossTab and the FactorPlot, we can straightforwardly conclude that survival for Women from Pclass1 is about 95-96%, as only 3 out of 94 Women from Pclass1 died.

It is apparent that regardless of P-class, Women were given priority while rescue. Even Men from Pclass1 have a very low survival rate.



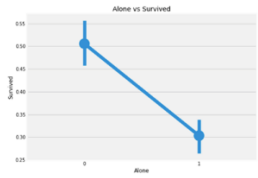
We can see that irrespective of PClass and Gender, chances of survival decreases as the age of the given passenger increases.

Gain Insight

We know that Age is a continuous numerical value. The continuous variable doesn't really work well with Machine Learning Variables when working with the other types of variables. Hence, we will categorize Age by diving the age range into a different decade. Hence people lying in each decade constitute one group. E.g. 0-10,10-20,20-30 etc.

Since we know that when you have a family with you, there are more helping hands with you to help you. Hence the family size is affecting the outcome, we should be looking at whether the passenger is with family or without. Hence, we will create an attribute column Is Alone which will be =1 if family size=0 otherwise it will be set to 0.

We can see that if a passenger has embarked onto the ship alone his chance of survival drops by a staggering 20%. Hence, it looks like an important feature as well.



Model Building

Since the output is binary, it is a type of Categorical Value in itself. We will have to use Classification algorithms in this case. We will use RandomForest, Decision Trees and Logistic Regress as well since the total number of possible outcomes are only 2.

After gathering relevant information about the dataset and their relation we build 5 models with different algorithms by hit and trial method to get the best combination

We build our first **Decision Tree Classifier** using attributes **Gender, Pclass** & Our **AgeCategory** which yields an **80.53% validation accuracy** on the training dataset and 68.4% testing accuracy, therefore, overfitting may not be there.

Next, we build another Decision Tree Classifier using **Gender, FareCategory,** and **PClass along with the predictions of decision tree classifier 1** which yields an **82.37% validation accuracy** on the training dataset and 70.4% testing accuracy. Thus, we can see that the accuracy of the total model has increased by just adding one nested stack in the model. When we would build the relation matrix for all the model, we would be able to see that these two algorithms are more alike as the second one is a stack of the first

Third, we build a Random Tree Classifier using **Gender, FareCategory** and **PClass** and **Family size** which gives us a slightly better **83% validation accuracy** on the training dataset and 64.4% testing accuracy, therefore, overfitting may be present as the difference between validation and testing accuracy is comparatively large. This means the randomForest classifier learned the training dataset instead of recognizing the actual patterns this would be later solved when we use Logistic regressing and stack them over each other.

Fourth we build a Random Tree Classifier using **Gender, AgeCategory and PClass and Family size** which gives us a slightly better **84% validation accuracy** on the training dataset and 68.3% testing accuracy, therefore, overfitting may be present. This means the randomForest classifier learned the training dataset instead of recognizing the actual patterns this would be later solved when we use Logistic regressing and stack them over each other. We also see that almost the same model with just a change in attribute from Farecategory to Agecategory increased the accuracy thus we can presume that the Age plays a more important role in determining if a person would survive or not rather than the Fare which validates our data analysis.

Lastly, we use a Logistic Regression Model using **gender, FamilySize, agecat, Pclass, Parch, SibSp, Fare** which gives us an accuracy of **79.8% validation accuracy** on the training dataset and 72.3% testing accuracy, therefore, overfitting is not present. It could be noted that if we used age instead of age-categorized the **validation accuracy was 93.4%** but the test accuracy swoop down to a low of 42.4% thus this was a severe case of overfitting. Through this case, we could conclude that creating categorized attributes decreased overfitting which is a problem of using small datasets.

We save the outputs of each prediction model which is simple 0s and 1s. If we concatenate all the prediction columns, we get 5 columns of 0s and 1s where each column is the independent prediction given by a model for a given passenger.

Using the 5 predictions as attributes we create 2^n = 2^5= 32 clusters. Now we calculate the total cases in a cluster that survived and that not survived. If the

number of survived in the cluster is more than the not survived, then the cluster is assigned the label survived i.e. value 1 and all the cases falling on that cluster is predicted as 1.

We create the 5 attribute columns using the 5 pre-trained models for the testing dataset and use the cluster from the trained model to predict the final score

We then use these outputs as features themselves for every passenger and count the chances of survival for a passenger given the predictions from each variable

Results when using age instead of categorized age

similarity matrix

models	1	2	3	4	5
1	x	x	x	x	x
2	0.96857	x	x	x	x
3	0.91	0.9281	x	x	x
4	0.852	0.873	0.869	x	x
5	0.92	0.93	0.92	0.84	x

Prediction 4	Prediction 3	Prediction 2	Prediction 1	Clusters no	survived	deceased
0	0	0	0	**0**	40	458
0	0	0	1	**1**	1	18
0	0	1	0	**2**	0	0
0	0	1	1	**3**	2	12
0	1	0	0	**4**	4	15
0	1	0	1	**5**	1	2
0	1	1	0	**6**	1	0
0	1	1	1	**7**	4	21
1	0	0	0	**8**		9
1	0	0	1	**9**	1	0
1	0	1	0	**10**	0	0
1	0	1	1	**11**	10	4
1	1	0	0	**12**	11	0
1	1	0	1	**13**	0	0
1	1	1	0	**14**	1	0
1	1	1	1	**15**	214	10

The prediction dataset shown above contains only 4 model results the cluster and its values are

CLUSTER AND THEIR RESPECTIVE PREDICTIONS

CLUSTERS	prediction	clusters	prediction
0	0	16	0
1	0	17	0
2	0	18	0
3	0	19	0
4	0	20	0
5	0	21	0
6	0	22	1
7	0	23	0
8	1	24	1
9	1	25	0
10	0	26	0
11	1	27	1
12	1	28	1
13	0	29	1
14	0	30	1
15	1	31	1

These are the cluster formed and the cluster prediction value which is calculated based on a weighted voting method

Now we observed that the value of the 5th model that is logistic regression showed cases of over-fitting thus the prediction of wrong values we changed the age which caused specificity of the dataset to categorize age which was more general/ created using clustering of age groups.

Upon doing this the results improved, and it was

similarity matrix

models	1	2	3	4	5
1	x	x	x	x	x
2	0.9685	x	x	x	x
3	0.9102	0.9281	x	x	x

4	0.938	0.95622	0.92255	x	x
5	0.9203	0.9248	0.91138	0.91132	x

As we can see the similarity of model 5 increased with the other model which is a good sign as other models had better validation accuracy.

clusters	survived	died	comment
0	70	446	70 survived irrespective of the fact that all models predicted death
1	1	13	13 died so the prediction value of the cluster 1 would be 0
2	0	0	
3	0	1	
4	7	6	
5	0	0	
6	0	0	
7	0	0	
8	0	3	
9	0	0	
10	0	0	
11	5	0	
12	3	0	
13	0	0	
14	0	0	
15	6	0	
16	11	18	1000 is the prediction set so the 5th model predicts that people would survive but the rest of them predicts that they would die. 11 survived but 18 died this is a very close difference so it could be further classified to increase validation accuracy
17	1	4	
18	0	0	
19	0	8	
20	4	8	
21	2	1	
22	2	0	
23	4	4	
24	0	0	
25	0	1	
26	0	0	
27	7	7	this case could further be classified to get better accuracy
28	1	1	
29	2	1	
30	0	0	
31	208	27	a clear case of survival with some outliers

clusters	prediction	clusters	prediction
0	0	**16**	0
1	0	**17**	0
2	0	**18**	0
3	0	**19**	0
4	1	**20**	0
5	0	**21**	1
6	0	**22**	1
7	0	**23**	0
8	1	**24**	0
9	0	**25**	0
10	0	**26**	0
11	1	**27**	0
12	1	**28**	0
13	0	**29**	1
14	0	**30**	0
15	1	**31**	1

Using the cluster value table we get the predictions of each cluster.

Results

After getting the outputs after successfully running the algorithm, we convert the DataFrame into a CSV

file and save it. The file is then submitted to Kaggle to get our Result.

We achieve a 77% accuracy in our very first iteration which lies in between our intended 75-85% range which clears it off any chances of over-fitting or under-fitting.

VII. COMPARATIVE STUDY / RESULTS AND DISCUSSION

We yielded a good score in terms of accuracy, 77% which clears us off the over-fitting trap. Upon using the Max Voting Ensemble Technique, we got a very high accuracy of 98.89% which was highly inclined to becoming a case of overfitting which is why we did not follow that path. Also, Precision itself cannot be one true metric to look at the performance of a model since accuracy takes in the assumption that the attributes are fairly distributed over the dataset which is hardly ever the case. Precision and Reminiscence should also be considered.

Upon dropping the XGBoost in the Average Voting case, the accuracy also fell to a mere 40% which is also clearly an instance of under-fitting; a poor performing model. Hence our model worked better in comparison with Simple Ensemble Techniques.

VIII. CONCLUSION AND FUTURE WORK

We have made a conclusion that our model shows no indication of over-fitting and displays potential in predicting the survival of a passenger up to a great extent. In terms of future work, we can work more deeply in the Feature Engineering to garner newer and more useful insights. We can also use more complex classifiers such as xgboost, AdaBoost and such to make our model better. In terms of our final model, we can fine-tune our model further by tweaking our polynomial equation and giving better models higher valued constant weights, like the weighted average technique.

IX. REFERENCES

1. Ali, K., & Pazzani, M. (1996). Error reduction through learning multiple descriptions. Machine Learning, 24, 173–202.
2. Alpaydin, E. (1993). Multiple networks for function learning. In Proceedings of the 1993 IEEE International Conference on Neural Networks, Vol. I, pp. 27–32 San Francisco.
3. Arbib, M. (Ed.). (1995). The Handbook of Brain Theory and Neural Networks. MIT Press.
4. Asker, L., & Maclin, R. (1997a). Ensembles as a sequence of classifiers. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 860–865 Nagoya, Japan.
5. Asker, L., & Maclin, R. (1997b). Feature engineering and classifier selection: A case study in Venusian volcano detection. In Proceedings of the Fourteenth International Conference on Machine Learning, pp. 3–11 Nashville, TN.
6. Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36, 105-139.
7. Baxt, W. (1992). Improving the accuracy of an artificial neural network using multiple differently trained networks. Neural Computation, 4, 772–780.
8. Breiman, L. (1996a). Bagging p edictors. Machine Learning, 24(2), 123–140.
9. https://pdfs.semanticscholar.org/1fe4/7722d5e65829c7e04b19648f39b22384d28c.pdf
10. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
11. https://pdfs.semanticscholar.org/53df/bc477ffb016e36d54f98e48695902d8eb566.pdf
12. http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/RN_11_02.pdf
13. https://ijarcce.com/wp-content/uploads/2015/02/IJARCCE3L.pdf