Kaggle Results Relative to Peers

Submission and Description	Public Score
titanic_xgboost.csv a few seconds ago by Anaswar Jayakumar MSDS 422 Classification Models - XGBoost	0.75837
titanic_naive_bayes.csv 23 minutes ago by Anaswar Jayakumar MSDS 422 Classification Models - Naive Bayes Classifier	0.75358
titanic_log_reg.csv an hour ago by Anaswar Jayakumar MSDS 422 Classification Models - Logistic Regression #3	0.75598
titanic_log_reg.csv an hour ago by Anaswar Jayakumar MSDS 422 Classification Models - Logistic Regression #2	0.75598
titanic_naive_bayes.csv 16 hours ago by Anaswar Jayakumar MSDS 422 Classification Models - Naive Bayes	0.74880
titanic_log_reg.csv 16 hours ago by Anaswar Jayakumar MSDS 422 Classification Models - Logistic Regression	0.76555

MSDS 422 Assignment 3

April 18, 2021

1 Assignment 3: Evaluating Classification Models

This week, you may be assigned one of two projects.

Compete in the Kaggle.com Titanic: Machine Learning through Disaster project located here (https://www.kaggle.com/c/titanic%20). You must make an account (free).

Use at least four binary (dichotomous) variables of your choice to build models. Predict the binary response variable of survival. Use cross-validation on the training set prior to submitting your forecasts to be graded on the Kaggle.com withheld test set. Employ two classification methods: (1) logistic regression as described in Chapter 4 of the Géron (2017) textbook and (2) naïve Bayes classification. Evaluate these methods within a cross-validation design as well as on the test set (minimum of two Kaggle.com submissions). Use the area under the receiver operating characteristic (ROC) curve as an index of classification as part of cross-validation.

Regarding the management problem, imagine that you are providing evidence regarding characteristics associated with survival on this ill-fated voyage to a historian writing a book. Which of the two modeling methods would you recommend and why?

For all Kaggle competitions, you must submit a screen snapshot that identifies you along with your scores on the submissions. Submit your work as a single pdf file that is legible. Include your code as an appendix. Look at the rubric to see how you will be graded. Your work will be compared against your peers on the performance metric(s).

Descriptive features: - PassengerId: Passenger's Id - Age: Age of the Passenger - Sex: Sex of the Passenger - Name: Name of the Passenger

Embarked: - Southampton - Cherbourg - Queenstown

Parch: Number of Parents/Children Aboard

SibSp: Number of Siblings/Spouses Aboard

Fare: Passenger Fare

Ticket: Ticket Number

Cabin: Cabin

Pclass: - 1 = 1st - 2 = 2nd - 3 = 3rd

Survived: - 1 for Survived - 0 for Not-Survived

2 Loading the required libraries and modules.

[1283]:
```
# main libraries
import os
import re
import pickle
import numpy as np
import pandas as pd

# visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.graph_objs as go
import plotly.io as pio
from plotly.subplots import make_subplots
import plotly.express as px
from plotly.offline import iplot, init_notebook_mode
import cufflinks as cf

# machine learning libraries:
from sklearn.model_selection import StratifiedKFold, cross_validate,␣
↪cross_val_score, train_test_split
from sklearn.preprocessing  import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import KNNImputer, IterativeImputer
from sklearn.ensemble import BaggingClassifier,␣
↪AdaBoostClassifier,GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")

# set some display options:
plt.rcParams['figure.dpi'] = 100
colors = px.colors.qualitative.Prism
pio.templates.default = "plotly_white"
```

[1284]:
```
# Initialize poly_features as True
poly_features = True
```

3 Loading the data and performing basic data checks.

3.1 Data Preproccessing

[2005]:
```
# import titanic data
titanic_train_df = pd.read_csv('/Users/anaswarjayakumar/Downloads/train (1).
↪csv')
titanic_test_df = pd.read_csv('/Users/anaswarjayakumar/Downloads/test (1).csv')

# show the head of the data
titanic_train_df.head()
```

[2005]:
```
   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 22.0000      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female 38.0000      1
2                             Heikkinen, Miss. Laina  female 26.0000      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female 35.0000      1
4                           Allen, Mr. William Henry    male 35.0000      0

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

[2006]:
```
titanic_test_df.head()
```

[2006]:
```
   PassengerId  Pclass                                          Name     Sex  \
0          892       3                              Kelly, Mr. James    male
1          893       3              Wilkes, Mrs. James (Ellen Needs)  female
2          894       2                     Myles, Mr. Thomas Francis    male
3          895       3                              Wirz, Mr. Albert    male
4          896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female

      Age  SibSp  Parch   Ticket    Fare Cabin Embarked
0 34.5000      0      0   330911  7.8292   NaN        Q
1 47.0000      1      0   363272  7.0000   NaN        S
2 62.0000      0      0   240276  9.6875   NaN        Q
3 27.0000      0      0   315154  8.6625   NaN        S
4 22.0000      1      1  3101298 12.2875   NaN        S
```

3.2 Data Exploration/Analysis

[1845]:
```python
# see information about the data
titanic_train_df.info()
print('_'*40)
titanic_test_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
________________________________________
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

[1846]:
```python
# show the types of columns
titanic_train_df.dtypes.to_frame().rename(columns={0:'Column type'})
```

[1846]:
```
              Column type
PassengerId        int64
Survived           int64
Pclass             int64
Name              object
Sex               object
Age              float64
SibSp              int64
Parch              int64
Ticket            object
Fare             float64
Cabin             object
Embarked          object
```

[1847]:
```
# finding the unique values in each column
for col in titanic_train_df.columns:
    print('We have {} unique values in {} column'.
 ↪format(len(titanic_train_df[col].unique()),col))
```

```
We have 891 unique values in PassengerId column
We have 2 unique values in Survived column
We have 3 unique values in Pclass column
We have 891 unique values in Name column
We have 2 unique values in Sex column
We have 89 unique values in Age column
We have 7 unique values in SibSp column
We have 7 unique values in Parch column
We have 681 unique values in Ticket column
We have 248 unique values in Fare column
We have 148 unique values in Cabin column
We have 4 unique values in Embarked column
```

[1848]:
```
titanic_train_df['SibSp'].unique()
```

[1848]:
```
array([1, 0, 3, 4, 2, 5, 8])
```

[1849]:
```
titanic_train_df['Parch'].unique()
```

[1849]:
```
array([0, 1, 2, 5, 3, 4, 6])
```

[1850]:
```
titanic_train_df['Embarked'].unique()
```

[1850]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[1851]:
```
titanic_train_df['Pclass'].unique()
```

[1851]:
```
array([3, 1, 2])
```

[1852]:
```python
titanic_train_df['Sex'].unique()
```

[1852]:
```
array(['male', 'female'], dtype=object)
```

[1853]:
```python
print('Age columns vary from {} to {}'.format(titanic_train_df['Age'].
↪min(),titanic_train_df['Age'].max()))
```

```
Age columns vary from 0.42 to 80.0
```

[1854]:
```python
# describe our data
titanic_train_df[titanic_train_df.select_dtypes(exclude='object').columns].
↪drop('PassengerId',axis=1).\
describe().style.background_gradient(axis=1,cmap=sns.light_palette('green',␣
↪as_cmap=True))
```

[1854]:
```
<pandas.io.formats.style.Styler at 0x7f81f95479a0>
```

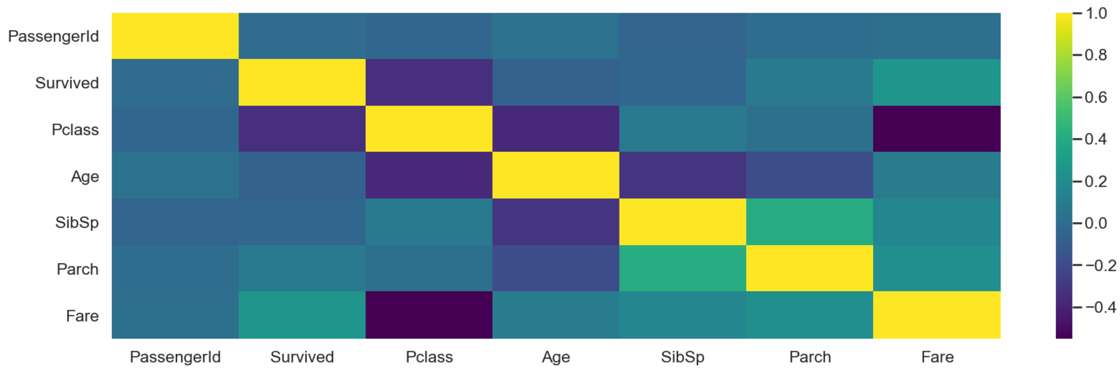
[1855]:
```python
# find the null values in each column
titanic_train_df.isnull().sum().to_frame().rename(columns={0:'Null values'})
```

[1855]:

	Null values
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2

[1856]:
```python
# Correlatation matrix for train data
plt.figure(figsize=(20,6));
sns.heatmap(titanic_train_df.corr(), cmap='viridis')
```

[1856]:
```
<AxesSubplot:>
```



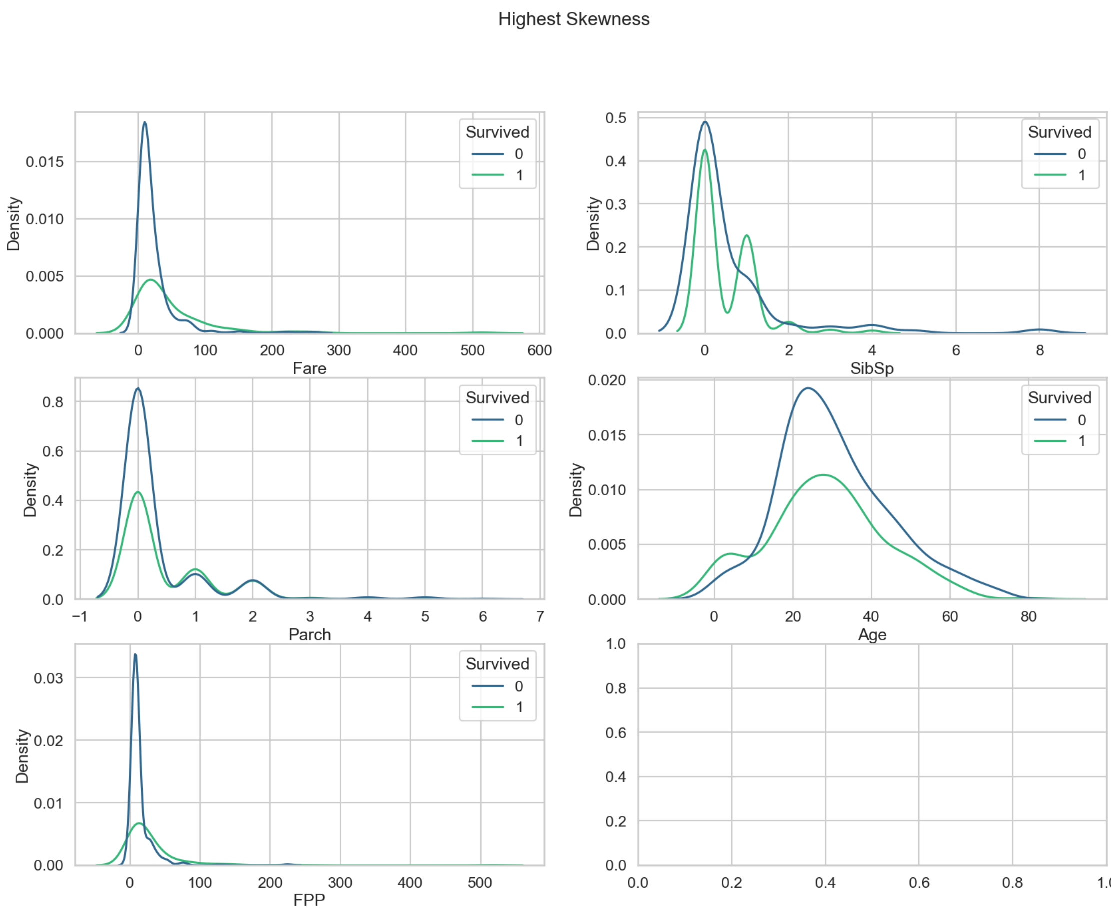
[1857]:
```
# lets take a look to the shape of columns
pd.set_option("display.float", "{:.4f}".format)
titanic_train_df.skew().to_frame().rename(columns={0:'Skewness'}).
 ↪sort_values('Skewness')
```

[1857]:

	Skewness
Pclass	-0.6305
PassengerId	0.0000
Age	0.3891
Survived	0.4785
Parch	2.7491
SibSp	3.6954
Fare	4.7873

[2007]:
```
titanic_train_df['FPP'] = (titanic_train_df['Fare']) \
                          / (1 + titanic_train_df['SibSp'] +␣
 ↪titanic_train_df['Parch'])
```

[2008]:
```
# Visualize columns have highest Skewness
fig, axes = plt.subplots(3,2, figsize=(20, 15));
fig.suptitle('Highest Skewness', fontsize=20);

sns.kdeplot(titanic_train_df['Fare'], ax=axes[0][0],␣
 ↪hue=titanic_train_df['Survived'], palette='viridis');
sns.kdeplot(titanic_train_df['SibSp'], ax=axes[0][1],␣
 ↪hue=titanic_train_df['Survived'], palette='viridis');
sns.kdeplot(titanic_train_df['Parch'], ax=axes[1][0],␣
 ↪hue=titanic_train_df['Survived'], palette='viridis');
sns.kdeplot(titanic_train_df['Age'], ax=axes[1][1],␣
 ↪hue=titanic_train_df['Survived'], palette='viridis');
sns.kdeplot(titanic_train_df['FPP'], ax=axes[2][0],␣
 ↪hue=titanic_train_df['Survived'], palette='viridis');
```



4 Setup and Basic EDA

4.0.1 Basic plotting functions

[1860]:
```
# use all data in visualization
df = pd.concat([titanic_train_df,titanic_test_df], axis=0)

# create a new column for the total number of family (Passenger )
df['family count']=df['Parch']+df['SibSp']+1

# capitalize sex column
df['Sex'] = df['Sex'].apply(lambda x:x.title())

# create a new column based on survived column (replace 1 with survived and 0␣
↪to not survived)
df['target'] = df['Survived'].map({1:'Survived',0:'Not Survived'})

# use columns with lowercases
```

```
df = df.rename(columns=lambda x:x.lower())
```

[1861]:
```
# lets define a function to plot a bar plot easily

def bar_plot(df,x,x_title,y,title,colors=None,text=None):
    fig = px.bar(x=x,
                 y=y,
                 text=text,
                 labels={"index": x_title}, # replaces default labels by column␣
↪name
                 data_frame=df,
                 color=colors,
                 barmode='group',
                 template="simple_white",
                 color_discrete_sequence=px.colors.qualitative.Prism)

    texts = [temp[col].values for col in y]
    for i, t in enumerate(texts):
        fig.data[i].text = t
        fig.data[i].textposition = 'inside'

    fig['layout'].title=title

    for trace in fig.data:
        trace.name = trace.name.replace('_',' ').capitalize()

    fig.update_yaxes(tickprefix="", showgrid=True)

    fig.show()
```

4.1 Univariate Visualisation - Categorical Features

4.1.1 Sex column

[1862]:
```
temp = pd.DataFrame()

for sex in pd.unique(df['sex']).tolist():
    temp[sex] = df[df['sex']==sex]['target'].value_counts()

temp = temp.rename(columns={0:'Female',1:'Male'}).T
temp['Total sum'] = temp.sum(axis=1)

bar_plot(temp.reset_index(),
         'index',
         'Age',
         ['Total sum','Survived','Not Survived'],
         title='Survived and Not-survived grouped by sex')
```

4.1.2 Pclass Column

[1863]:
```
temp = pd.DataFrame()

for p in set(pd.unique(df['pclass'])):
    temp[p] = df[df['pclass']==p]['target'].value_counts()

temp = temp.rename(columns={1:'Class 1',2:'Class 2', 3:'Class 3'}).T
temp['Total sum'] = temp.sum(axis=1)

bar_plot(temp.reset_index(),
         'index',
         'Pclass',
         ['Total sum','Survived','Not Survived'],
         title='Survived and Not-survived grouped by Pclass')
```

4.1.3 Family Count Column

[1864]:
```
# before applying particular test we have to look for Contingency table
family_count = pd.crosstab(index=df['family count'],columns=df['target'])
family_count
```

[1864]:

target family count	Not Survived	Survived
1	374	163
2	72	89
3	43	59
4	8	21
5	12	3
6	19	3
7	8	4
8	6	0
11	7	0

[1865]:
```
temp = pd.crosstab(index=df['family count'],columns=df['target']).reset_index()

temp['Total sum'] = temp[['Not Survived', 'Survived']].sum(axis=1)

bar_plot(temp,
         'family count',
         'Family number',
         ['Total sum','Survived','Not Survived'],
         title='Survived and Not-survived grouped by Family number')
```

4.1.4 Embarked Count column

[1866]:
```
df['embarked'].value_counts().to_frame().rename(columns={'embarked':'Total␣
↪Count'})
```

[1866]:
```
   Total Count
S          914
C          270
Q          123
```

[1867]:
```
# we are still using the whole data for visualizion but only train_df is␣
↪counted because test_df doesn't
# have Survived column
temp = pd.DataFrame()

for e in df['embarked'].unique().tolist():
    temp[e] = df[df['embarked']==e]['target'].value_counts()

temp = temp.T.rename(index={'S':'Southampton','C':'Cherbourg','Q':'Queenstown'})
temp['Total sum'] = temp.sum(axis=1)

bar_plot(temp.reset_index(),
         'index',
         'Embarked',
         ['Total sum','Survived','Not Survived'],
         title='Survived and Not-survived grouped by Embarked column')
```

4.2 Univariate Visualisation - Numerical Features

4.2.1 Age Column

[2067]:
```
df['age_category'] = pd.cut(df['age'].fillna(df['age'].mean()).astype(int),
                            bins=[-1,5,10,15,20,25,30,40,50,60,100],
                                                                        ␣
↪labels=["<=5","5-10","10-15","15-20","20-25","25-30","30-40","40-50","50-60",
                                    ">=60"])

temp = pd.DataFrame()
for age in df['age_category'].unique().tolist():
    temp[age] = df[df['age_category']==age]['target'].value_counts()

temp = temp.T.reset_index()
temp['Total sum'] = temp.sum(axis=1)

bar_plot(temp.reset_index(),
         'index',
         'Age Category',
         ['Total sum','Survived','Not Survived'],
```

```
        title='Survived and Not-survived grouped by Age column')


fig = make_subplots(rows=2, cols=2,
                    specs=[[{"colspan": 2}, None],
                           [{}, {}]],
                    subplot_titles=('Age distribution',
                                    'Survived',
                                    'Not Survived'))

fig.add_trace(go.Histogram(x=df['age']),
              row=1, col=1)

fig.add_trace(go.Histogram(x=df[df['target']=='Survived']['age']),
              row=2, col=1)

fig.add_trace(go.Histogram(x=df[df['target']=='Not Survived']['age']),
              row=2, col=2)

fig.update_layout(showlegend=False, title_text='Distribution for Age')
fig.show()
```

4.2.2 Fare Column

[2065]:
```
df['fare_category'] = pd.cut(df['fare'].fillna(df['fare'].mean()).astype(int),
                             bins=[-1,5,15,25,35,45,55,65,100,1000],
                            ␣
 ↪labels=["<=5","5-15","15-25","25-35","35-45","45-55","55-65","65-100",">=100"])

temp = pd.DataFrame()
for age in df['fare_category'].unique().tolist():
    temp[age] = df[df['fare_category']==age]['target'].value_counts()

temp = temp.T.reset_index()
temp['Total sum'] = temp.sum(axis=1)

bar_plot(temp.reset_index(),
         'index',
         'Fare Category',
         ['Total sum','Survived','Not Survived'],
         title='Survived and Not-survived grouped by Fare column')

fig = make_subplots(rows=2, cols=2,
                    specs=[[{"colspan": 2}, None],
                           [{}, {}]],
                    subplot_titles=('Fare distribution',
                                    'Survived',
```

```
                                    'Not Survived'))

fig.add_trace(go.Histogram(x=df['fare'][:len(titanic_train_df)]),
              row=1, col=1)

fig.add_trace(go.Histogram(x=df[df['target']=='Survived']['fare'][:
 ↪len(titanic_train_df)]),
              row=2, col=1)

fig.add_trace(go.Histogram(x=df[df['target']=='Not Survived']['fare'][:
 ↪len(titanic_train_df)]),
              row=2, col=2)

fig.update_layout(showlegend=False, title_text='Distribution for Fare')
fig.show()
```

4.3 Multivariate Visualization

4.3.1 Multi-Box plots for each column

[1870]:
```
#create a function to plot multi box plots easily

def multi_box(df,cat_col,dist_col,color_col):

    y = []
    x = []

    if len(df[color_col].unique())!= 2:
        return 'Maximun number of unique values in the color columns is 2'

    for c in set(df[cat_col].unique().tolist()):
        for t in set(df[color_col].unique()):
            y.append(df[(df[cat_col]==c) & (df[color_col]==t)][dist_col].values)
            x.append(str(c)+' ('+str(t)+')')

    colors = ['rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)',
              'rgba(251, 43, 43, 0.5)', 'rgba(125, 251, 137, 0.5)']

    traces = []

    for xd, yd, cls in zip(x, y, colors[:2*len(df[cat_col].unique())]):
            traces.append(go.Box(y=yd,
                                 name=xd,
```

```
                                    boxpoints='all',
                                    jitter=0.5,
                                    whiskerwidth=0.2,
                                    fillcolor=cls,
                                    marker=dict(size=2),
                                    line=dict(width=1)))

    layout = go.Layout(title='{} distribution colored by {} grouped by {}'.
↪format(dist_col.title(),
                                                                                    ␣
↪ color_col.title(),
                                                                                    ␣
↪ cat_col.title()),
        xaxis=dict(title=cat_col,
                   titlefont=dict(size=16)),

        yaxis=dict(title='Distribution',
                   autorange=True,
                   showgrid=True,
                   zeroline=True,
                   dtick=5,
                   gridcolor='rgb(255, 255, 255)',
                   gridwidth=1,
                   zerolinecolor='rgb(255, 255, 255)',
                   zerolinewidth=2,
                   titlefont=dict(
                   size=16)),

        margin=dict(l=40,
                    r=30,
                    b=80,
                    t=100),

        paper_bgcolor='rgb(255, 255, 255)',
        plot_bgcolor='rgb(255, 243, 192)',
        showlegend=False)

    fig = go.Figure(data=traces, layout=layout)
    iplot(fig)
```

4.3.2 Age distribution for Pclass column

[1871]:
```
multi_box(df.dropna(),'pclass','age','target')
```

[1872]:
```
multi_box(df.dropna(),'sex','age','target')
```

[1873]:
```
# Create dataframe for categorical variables of training
# cat_df = titanic_train_df.select_dtypes(include = ["object"])
# cat_df = cat_df.drop(columns = ['Name', 'Ticket', 'Cabin', 'target'])

# Create dataframe for continuous variables and drop PassengerId column
# cont_df = titanic_train_df.select_dtypes(include = ["float64", "int64"])
# cont_df = cont_df.drop(columns = ['PassengerId', 'Survived'])

# titanic_train_df = titanic_train_df.drop(columns = ['Name', 'Ticket',␣
↪'Cabin'])
# titanic_test_df = titanic_test_df.drop(columns = ['Name', 'Ticket', 'Cabin'])

cat_cols = ['Sex', 'Embarked']
cont_cols = ['Age', 'Fare', 'SibSp', 'Parch', 'Pclass']
```

[1874]:
```
cat_cols
```

[1874]:
```
['Sex', 'Embarked']
```

[1875]:
```
cont_cols
```

[1875]:
```
['Age', 'Fare', 'SibSp', 'Parch', 'Pclass']
```

[1877]:
```
del test_cont_df
```

[1878]:
```
# Create dataframe for continuous variables and drop PassengerId columns in␣
↪test data
# test_cont_df = titanic_test_df.select_dtypes(include = ["float64", "int64"])
# test_cont_df = test_cont_df.drop(columns = ['PassengerId'])
# print(test_cont_df.shape)
```

5 Setup and Basic EDA Part II

5.1 Label Encoding

[2009]:
```
titanic_train_df[cat_cols]
```

[2009]:
```
        Sex Embarked
0      male        S
1    female        C
2    female        S
3    female        S
4      male        S
..      …       …
886    male        S
887  female        S
888  female        S
```

```
889    male        C
890    male        Q

[891 rows x 2 columns]
```

[2010]:
```python
titanic_train_df['Embarked'].unique().tolist()
```

[2010]:
```
['S', 'C', 'Q', nan]
```

[2011]:
```python
titanic_train_df['Embarked'].fillna('N', inplace = True)
titanic_train_df
```

[2011]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …       …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex      Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0000      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female 38.0000      1
2                               Heikkinen, Miss. Laina  female  26.0000      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0000      1
4                             Allen, Mr. William Henry    male  35.0000      0
..                                                 …     …       …    …
886                              Montvila, Rev. Juozas    male  27.0000      0
887                       Graham, Miss. Margaret Edith  female  19.0000      0
888           Johnston, Miss. Catherine Helen "Carrie"  female      nan      1
889                              Behr, Mr. Karl Howell    male  26.0000      0
890                                Dooley, Mr. Patrick    male  32.0000      0

     Parch            Ticket    Fare Cabin Embarked     FPP
0        0         A/5 21171  7.2500   NaN        S  3.6250
1        0          PC 17599 71.2833   C85        C 35.6416
2        0  STON/O2. 3101282  7.9250   NaN        S  7.9250
3        0            113803 53.1000  C123        S 26.5500
4        0            373450  8.0500   NaN        S  8.0500
..     …               …     …   …       …
886      0            211536 13.0000   NaN        S 13.0000
887      0            112053 30.0000   B42        S 30.0000
```

```
888       2        W./C. 6607 23.4500    NaN        S  5.8625
889       0            111369 30.0000   C148        C 30.0000
890       0            370376  7.7500    NaN        Q  7.7500

[891 rows x 13 columns]
```

[2012]:
```python
titanic_train_df['Embarked'].isnull().sum()
```

[2012]: 0

[2013]:
```python
titanic_test_df['Embarked'].fillna('N', inplace = True)
titanic_test_df
```

[2013]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …      …                                           …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex     Age  SibSp  Parch              Ticket      Fare Cabin Embarked
0      male 34.5000      0      0              330911    7.8292   NaN        Q
1    female 47.0000      1      0              363272    7.0000   NaN        S
2      male 62.0000      0      0              240276    9.6875   NaN        Q
3      male 27.0000      0      0              315154    8.6625   NaN        S
4    female 22.0000      1      1             3101298   12.2875   NaN        S
..      …     …    …    …                 …       …   …      …
413    male     nan      0      0           A.5. 3236    8.0500   NaN        S
414  female 39.0000      0      0            PC 17758  108.9000  C105        C
415    male 38.5000      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416    male     nan      0      0              359309    8.0500   NaN        S
417    male     nan      1      1                2668   22.3583   NaN        C

[418 rows x 11 columns]
```

[2014]:
```python
titanic_test_df['Embarked'].isnull().sum()
```

[2014]: 0

[2015]:
```python
titanic_train_df['Age'] = titanic_train_df['Age'].fillna(titanic_train_df.
 ↪groupby('Sex')['Age'].transform('mean'))
```

```
titanic_train_df
```

```
[2015]:      PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex     Age  SibSp  \
0                              Braund, Mr. Owen Harris    male 22.0000      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female 38.0000      1
2                               Heikkinen, Miss. Laina  female 26.0000      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female 35.0000      1
4                             Allen, Mr. William Henry    male 35.0000      0
..                                                 ...     ...     ...    ...
886                              Montvila, Rev. Juozas    male 27.0000      0
887                       Graham, Miss. Margaret Edith  female 19.0000      0
888           Johnston, Miss. Catherine Helen "Carrie"  female 27.9157      1
889                              Behr, Mr. Karl Howell    male 26.0000      0
890                                Dooley, Mr. Patrick    male 32.0000      0

     Parch            Ticket    Fare Cabin Embarked     FPP
0        0         A/5 21171  7.2500   NaN        S  3.6250
1        0          PC 17599 71.2833   C85        C 35.6416
2        0  STON/O2. 3101282  7.9250   NaN        S  7.9250
3        0            113803 53.1000  C123        S 26.5500
4        0            373450  8.0500   NaN        S  8.0500
..     ...               ...     ...   ...      ...
886      0            211536 13.0000   NaN        S 13.0000
887      0            112053 30.0000   B42        S 30.0000
888      2        W./C. 6607 23.4500   NaN        S  5.8625
889      0            111369 30.0000  C148        C 30.0000
890      0            370376  7.7500   NaN        Q  7.7500

[891 rows x 13 columns]
```

```
[2016]: titanic_train_df['Age'].isnull().sum()
```

```
[2016]: 0
```

[2017]:
```
titanic_test_df['Age'] = titanic_test_df['Age'].fillna(titanic_test_df.
    ↪groupby('Sex')['Age'].transform('mean'))
titanic_test_df
```

[2017]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …       …                                           …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex      Age  SibSp  Parch              Ticket      Fare Cabin Embarked
0      male  34.5000      0      0              330911    7.8292   NaN        Q
1    female  47.0000      1      0              363272    7.0000   NaN        S
2      male  62.0000      0      0              240276    9.6875   NaN        Q
3      male  27.0000      0      0              315154    8.6625   NaN        S
4    female  22.0000      1      1             3101298   12.2875   NaN        S
..      …        …    …     …                 …       …   …      …
413    male  30.2727      0      0           A.5. 3236    8.0500   NaN        S
414  female  39.0000      0      0            PC 17758  108.9000  C105        C
415    male  38.5000      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416    male  30.2727      0      0              359309    8.0500   NaN        S
417    male  30.2727      1      1                2668   22.3583   NaN        C

[418 rows x 11 columns]
```

[2018]:
```
titanic_test_df['Age'].isnull().sum()
```

[2018]: 0

[2019]:
```
titanic_train_df['Fare'].isnull().sum()
```

[2019]: 0

[2020]:
```
titanic_test_df['Fare'].isnull().sum()
```

[2020]: 1

[2021]:
```
titanic_test_df['Fare'] = titanic_test_df['Fare'].fillna(titanic_test_df.
    ↪groupby('Pclass')['Fare'].\
                                                        transform('mean'))
titanic_test_df
```

[2021]:
```
     PassengerId  Pclass                                          Name  \
0            892       3                              Kelly, Mr. James
1            893       3              Wilkes, Mrs. James (Ellen Needs)
2            894       2                     Myles, Mr. Thomas Francis
3            895       3                              Wirz, Mr. Albert
4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
..           …       …                                             …
413         1305       3                            Spector, Mr. Woolf
414         1306       1                  Oliva y Ocana, Dona. Fermina
415         1307       3                  Saether, Mr. Simon Sivertsen
416         1308       3                           Ware, Mr. Frederick
417         1309       3                      Peter, Master. Michael J

        Sex      Age  SibSp  Parch              Ticket      Fare Cabin Embarked
0      male  34.5000      0      0              330911    7.8292   NaN        Q
1    female  47.0000      1      0              363272    7.0000   NaN        S
2      male  62.0000      0      0              240276    9.6875   NaN        Q
3      male  27.0000      0      0              315154    8.6625   NaN        S
4    female  22.0000      1      1             3101298   12.2875   NaN        S
..      …       …     …    …                 …       …   …      …
413    male  30.2727      0      0           A.5. 3236    8.0500   NaN        S
414  female  39.0000      0      0            PC 17758  108.9000  C105        C
415    male  38.5000      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
416    male  30.2727      0      0              359309    8.0500   NaN        S
417    male  30.2727      1      1                2668   22.3583   NaN        C

[418 rows x 11 columns]
```

[2022]:
```
titanic_test_df['Fare'].isnull().sum()
```

[2022]:
```
0
```

[2023]:
```
# Import LabelEncoder from sklearn.preprocessing
from sklearn.preprocessing import LabelEncoder

# Iterate through each category column and convert to numeric using␣
↪LabelEncoder. Then transform the column
# and assign back to the original column
for key in cat_cols:
    le = LabelEncoder()
    labels = list(titanic_train_df[key].unique())
    labels += list(titanic_test_df[key].unique())

    # Create mapping from labels to integers
    le.fit(labels)
    # Transform the train and test consistently
    titanic_train_df[key] = le.transform(titanic_train_df[key])
```

```
    titanic_test_df[key] = le.transform(titanic_test_df[key])
```

5.2 Correlation between continuous columns and survived

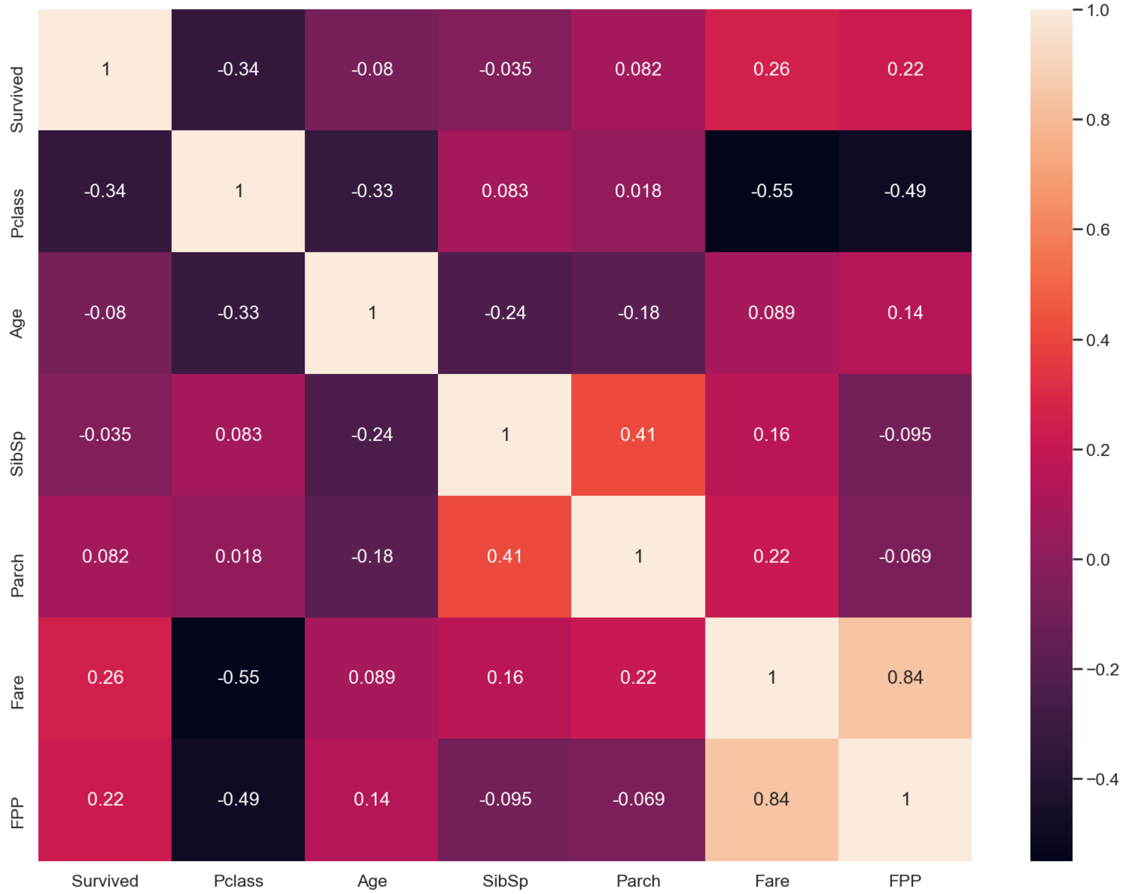
[2024]:
```
titanic_train_df.head()
```

[2024]:
```
   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    1 22.0000      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…    0 38.0000      1
2                             Heikkinen, Miss. Laina    0 26.0000      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    0 35.0000      1
4                           Allen, Mr. William Henry    1 35.0000      0

   Parch            Ticket    Fare Cabin  Embarked     FPP
0      0         A/5 21171  7.2500   NaN         3  3.6250
1      0          PC 17599 71.2833   C85         0 35.6416
2      0  STON/O2. 3101282  7.9250   NaN         3  7.9250
3      0            113803 53.1000  C123         3 26.5500
4      0            373450  8.0500   NaN         3  8.0500
```

[2025]:
```
corr_df = titanic_train_df.copy()
corr_df.drop(columns = cat_cols, inplace = True)
corr_df.drop(columns = ['Name', 'Ticket', 'Cabin'], inplace = True)
corr_df.drop(columns = ["PassengerId"], inplace = True)
corr_df
```

[2025]:
```
     Survived  Pclass     Age  SibSp  Parch    Fare     FPP
0           0       3 22.0000      1      0  7.2500  3.6250
1           1       1 38.0000      1      0 71.2833 35.6416
2           1       3 26.0000      0      0  7.9250  7.9250
3           1       1 35.0000      1      0 53.1000 26.5500
4           0       3 35.0000      0      0  8.0500  8.0500
..        …       …       …      …      …       …       …
886         0       2 27.0000      0      0 13.0000 13.0000
887         1       1 19.0000      0      0 30.0000 30.0000
888         0       3 27.9157      1      2 23.4500  5.8625
889         1       1 26.0000      0      0 30.0000 30.0000
890         0       3 32.0000      0      0  7.7500  7.7500

[891 rows x 7 columns]
```

[2026]:
```
import seaborn as sns
plt.subplots(figsize=(20,15))
sns.heatmap(corr_df.corr(), annot = True)
plt.show()
```



5.3 Correlation between significant categorical columns and survived

[2027]:
```
# correlation between columns and target column
corr = titanic_train_df.corr()
corr['Survived'].sort_values(ascending=False)[1:].to_frame()\
.style.background_gradient(axis=1,cmap=sns.light_palette('green', as_cmap=True))
```

[2027]: <pandas.io.formats.style.Styler at 0x7f81e9c59ee0>

6 Creating arrays for the features and the response variable.

[2028]:
```
# Take the loss column and set it as the target column since the loss variable␣
↪is what is being predicted
target_column = ['Survived']
# Create the list of predictors variables
predictors = ['Fare', 'Age', 'Parch', 'SibSp', 'Embarked', 'Pclass', 'Sex']
```

6.1 Normalizing predictor columns

[2029]:
```
# Import MinMaxScaler from sklearn.preprocessing and ColumnTransformer from␣
↪sklearn.compose
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer

# Normalize the values in the columns of the categorical dataframe
minmax_transformer = Pipeline(steps=[('minmax', MinMaxScaler())])
standard_transformer = Pipeline(steps=[('standard', StandardScaler())])
preprocessor = ColumnTransformer(
        remainder='passthrough', # passthough features not listed
        transformers=[('ss', standard_transformer, predictors)])

preprocessor.fit(titanic_train_df[predictors])

# Create an array containing the normalized values for both the train and the␣
↪test
norm_train = preprocessor.transform(titanic_train_df[predictors])
norm_test = preprocessor.transform(titanic_test_df[predictors])
```

[2030]:
```
print(norm_train.shape)
```

```
(891, 7)
```

[2031]:
```
norm_train = np.c_[norm_train, titanic_train_df['Survived'].values]
print(norm_train.shape)
```

```
(891, 8)
```

[2032]:
```
print(norm_test.shape)
```

```
(418, 7)
```

[2033]:
```
titanic_train_df['Survived'].values.shape
```

[2033]:
```
(891,)
```

[2034]:
```
# Convert the array containing the normalized values to a dataframe
train_numeric_df = pd.DataFrame(data = norm_train, index = titanic_train_df.
 ↪index,
                                 columns = predictors + ['Survived'])
print(train_numeric_df)

print("\n")

test_numeric_df = pd.DataFrame(data = norm_test, index = titanic_test_df.index,␣
 ↪columns = predictors)
print(test_numeric_df)
```

```
        Fare     Age   Parch   SibSp  Embarked  Pclass     Sex  Survived
0    -0.5024 -0.5947 -0.4737  0.4328    0.5627  0.8274  0.7377    0.0000
1     0.7868  0.6353 -0.4737  0.4328   -2.0085 -1.5661 -1.3556    1.0000
2    -0.4889 -0.2872 -0.4737 -0.4745    0.5627  0.8274 -1.3556    1.0000
3     0.4207  0.4047 -0.4737  0.4328    0.5627 -1.5661 -1.3556    1.0000
4    -0.4863  0.4047 -0.4737 -0.4745    0.5627  0.8274  0.7377    0.0000
..       …       …       …       …         …       …       …       …
886 -0.3867 -0.2103 -0.4737 -0.4745    0.5627 -0.3694  0.7377    0.0000
887 -0.0444 -0.8254 -0.4737 -0.4745    0.5627 -1.5661 -1.3556    1.0000
888 -0.1763 -0.1399  2.0089  0.4328    0.5627  0.8274 -1.3556    0.0000
889 -0.0444 -0.2872 -0.4737 -0.4745   -2.0085 -1.5661  0.7377    1.0000
890 -0.4924  0.1740 -0.4737 -0.4745   -0.2944  0.8274  0.7377    0.0000

[891 rows x 8 columns]


        Fare     Age   Parch   SibSp  Embarked  Pclass     Sex
0    -0.4908  0.3662 -0.4737 -0.4745   -0.2944  0.8274  0.7377
1    -0.5075  1.3272 -0.4737  0.4328    0.5627  0.8274 -1.3556
2    -0.4534  2.4804 -0.4737 -0.4745   -0.2944 -0.3694  0.7377
3    -0.4740 -0.2103 -0.4737 -0.4745    0.5627  0.8274  0.7377
4    -0.4010 -0.5947  0.7676  0.4328    0.5627  0.8274 -1.3556
..       …       …       …       …         …       …       …
413 -0.4863  0.0413 -0.4737 -0.4745    0.5627  0.8274  0.7377
414  1.5442  0.7122 -0.4737 -0.4745   -2.0085 -1.5661 -1.3556
415 -0.5024  0.6738 -0.4737 -0.4745    0.5627  0.8274  0.7377
416 -0.4863  0.0413 -0.4737 -0.4745    0.5627  0.8274  0.7377
417 -0.1982  0.0413  0.7676  0.4328   -2.0085  0.8274  0.7377

[418 rows x 7 columns]
```

7 Add Bias

[2035]:
```
train_numeric_df.insert(0, "bias", 1)
print(train_numeric_df)
print(train_numeric_df.dtypes)
```

```
     bias    Fare     Age   Parch   SibSp  Embarked  Pclass     Sex  Survived
0       1 -0.5024 -0.5947 -0.4737  0.4328    0.5627  0.8274  0.7377    0.0000
1       1  0.7868  0.6353 -0.4737  0.4328   -2.0085 -1.5661 -1.3556    1.0000
2       1 -0.4889 -0.2872 -0.4737 -0.4745    0.5627  0.8274 -1.3556    1.0000
3       1  0.4207  0.4047 -0.4737  0.4328    0.5627 -1.5661 -1.3556    1.0000
4       1 -0.4863  0.4047 -0.4737 -0.4745    0.5627  0.8274  0.7377    0.0000
..    …      …      …      …      …       …      …      …       …
886     1 -0.3867 -0.2103 -0.4737 -0.4745    0.5627 -0.3694  0.7377    0.0000
887     1 -0.0444 -0.8254 -0.4737 -0.4745    0.5627 -1.5661 -1.3556    1.0000
888     1 -0.1763 -0.1399  2.0089  0.4328    0.5627  0.8274 -1.3556    0.0000
889     1 -0.0444 -0.2872 -0.4737 -0.4745   -2.0085 -1.5661  0.7377    1.0000
890     1 -0.4924  0.1740 -0.4737 -0.4745   -0.2944  0.8274  0.7377    0.0000

[891 rows x 9 columns]
bias          int64
Fare        float64
Age         float64
Parch       float64
SibSp       float64
Embarked    float64
Pclass      float64
Sex         float64
Survived    float64
dtype: object
```

[2036]:
```
test_numeric_df.insert(0, "bias", 1)
print(test_numeric_df)
print(test_numeric_df.dtypes)
```

```
     bias    Fare     Age   Parch   SibSp  Embarked  Pclass     Sex
0       1 -0.4908  0.3662 -0.4737 -0.4745   -0.2944  0.8274  0.7377
1       1 -0.5075  1.3272 -0.4737  0.4328    0.5627  0.8274 -1.3556
2       1 -0.4534  2.4804 -0.4737 -0.4745   -0.2944 -0.3694  0.7377
3       1 -0.4740 -0.2103 -0.4737 -0.4745    0.5627  0.8274  0.7377
4       1 -0.4010 -0.5947  0.7676  0.4328    0.5627  0.8274 -1.3556
..    …      …      …      …      …       …      …      …
413     1 -0.4863  0.0413 -0.4737 -0.4745    0.5627  0.8274  0.7377
414     1  1.5442  0.7122 -0.4737 -0.4745   -2.0085 -1.5661 -1.3556
415     1 -0.5024  0.6738 -0.4737 -0.4745    0.5627  0.8274  0.7377
416     1 -0.4863  0.0413 -0.4737 -0.4745    0.5627  0.8274  0.7377
417     1 -0.1982  0.0413  0.7676  0.4328   -2.0085  0.8274  0.7377
```

```
[418 rows x 8 columns]
bias          int64
Fare        float64
Age         float64
Parch       float64
SibSp       float64
Embarked    float64
Pclass      float64
Sex         float64
dtype: object
```

8 Creating the Training and Test Datasets

[2037]:
```
train_numeric_df.describe()
```

[2037]:
```
           bias      Fare       Age     Parch     SibSp  Embarked    Pclass  \
count  891.0000  891.0000  891.0000  891.0000  891.0000  891.0000  891.0000
mean     1.0000   -0.0000   -0.0000    0.0000    0.0000    0.0000   -0.0000
std      0.0000    1.0006    1.0006    1.0006    1.0006    1.0006    1.0006
min      1.0000   -0.6484   -2.2538   -0.4737   -0.4745   -2.0085   -1.5661
25%      1.0000   -0.4891   -0.5947   -0.4737   -0.4745   -0.2944   -0.3694
50%      1.0000   -0.3574    0.0203   -0.4737   -0.4745    0.5627    0.8274
75%      1.0000   -0.0242    0.4047   -0.4737    0.4328    0.5627    0.8274
max      1.0000    9.6672    3.8642    6.9741    6.7842    0.5627    0.8274

            Sex  Survived
count  891.0000  891.0000
mean    -0.0000    0.3838
std      1.0006    0.4866
min     -1.3556    0.0000
25%     -1.3556    0.0000
50%      0.7377    0.0000
75%      0.7377    1.0000
max      0.7377    1.0000
```

8.1 Polynomial Features for Training Set

[2038]:
```
titanic_train_df['Age'].isnull().sum()
```

[2038]:
```
0
```

[2039]:
```
from sklearn.preprocessing import PolynomialFeatures

# If poly_features is True, create the polynomial features for the continuous␣
↪variables in the training set
if poly_features:
    poly_features_cont = PolynomialFeatures(degree = 2, include_bias = False)
```

```
    # Fit the polynomial features
    poly_features_cont.fit(titanic_train_df[cont_cols].values)

    X_poly_train_cont = poly_features_cont.
 ↪transform(titanic_train_df[cont_cols].values)
```

[2040]:
```
if poly_features:
    print(X_poly_train_cont.shape)
```

```
(891, 20)
```

[2041]:
```
np.count_nonzero(np.isnan(X_poly_train_cont))
```

[2041]: 0

[2042]:
```
np.count_nonzero(np.isnan(train_numeric_df[predictors].values))
```

[2042]: 0

[2043]:
```
"PassengerId" in predictors
```

[2043]: False

[2044]:
```
predictors_with_bias = ['bias'] + predictors
predictors_with_bias
```

[2044]: ['bias', 'Fare', 'Age', 'Parch', 'SibSp', 'Embarked', 'Pclass', 'Sex']

[2045]:
```
len(predictors_with_bias)
```

[2045]: 8

[2112]:
```
if poly_features:
    X = np.c_[train_numeric_df[predictors_with_bias].values, X_poly_train_cont]
else:
    X = train_numeric_df[predictors_with_bias].values

y = train_numeric_df[target_column].values

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.20)
print(X_train.shape)
print(X_val.shape)
```

```
(712, 28)
(179, 28)
```

8.2 Polynomial Features for Test Set

[2113]:
```
titanic_test_df['Age'].isnull().sum()
```

[2113]: 0

[2114]:
```
titanic_test_df['Fare'].isnull().sum()
```

[2114]: 0

[2115]:
```
if poly_features:
    X_poly_test_cont = poly_features_cont.transform(titanic_test_df[cont_cols].
 ↪values)
```

[2116]:
```
if poly_features:
    print(X_poly_test_cont.shape)
```

```
(418, 20)
```

[2117]:
```
if poly_features:
    X_submission = np.c_[test_numeric_df[predictors_with_bias].values,␣
 ↪X_poly_test_cont]
else:
    X_submission = test_numeric_df[predictors_with_bias].values
```

[2118]:
```
X_submission.shape
```

[2118]: (418, 28)

[2119]:
```
np.count_nonzero(np.isnan(X))
```

[2119]: 0

9 Build, Predict and Evaluate the Classification Models

9.1 Logistic Regression

[2306]:
```
from sklearn.model_selection import GridSearchCV

param_lr={'penalty':['l1','l2'],
         'C' : [0.01,0.1,1,10,50,100,200,300],
         'solver':['liblinear', 'saga', 'lbfgs']}

gs_lr = GridSearchCV(LogisticRegression(),param_grid = param_lr,␣
 ↪scoring="accuracy",n_jobs=-1)
gs_lr.fit(X_train, y_train)
best_lr=gs_lr.best_estimator_
print(best_lr)
```

```
print('score=',gs_lr.best_score_)
```

```
LogisticRegression(C=0.1)
score= 0.8005811090318133
```

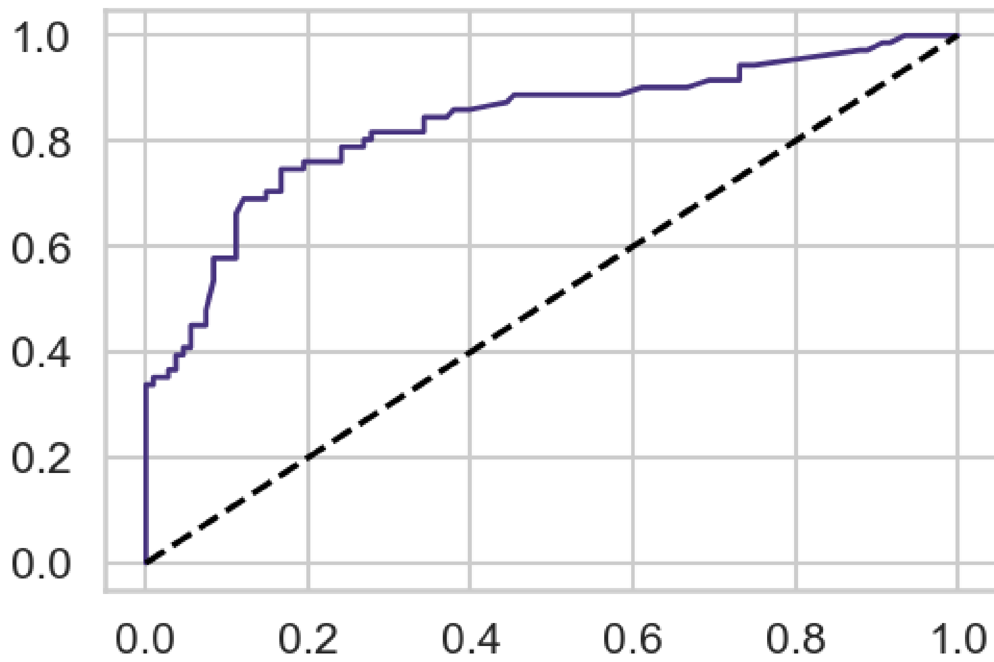
[2307]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc

log_reg = LogisticRegression(C=300, penalty='l1', solver='liblinear')
y_score = log_reg.fit(X_train, y_train).decision_function(X_val)
fpr, tpr, thresholds = roc_curve(y_val, y_score)
roc_auc = auc(fpr, tpr)
roc_auc
```

[2307]:
```
0.8319640062597811
```

[2308]:
```
thresholds
```

[2308]:
```
array([ 5.29097168,  4.29097168,  4.20108261,  3.94811018,  3.82644451,
        1.77258328,  1.74025326,  1.55187411,  1.51569313,  1.50509158,
        1.46979484,  1.38968298,  1.37990577,  1.36808916,  1.30823108,
        1.16189927,  1.00863197,  0.96200529,  0.90589336,  0.82733423,
        0.70892805,  0.60629424,  0.59362426,  0.51255422,  0.40943374,
        0.21440223,  0.20033158,  0.16744473, -0.04808777, -0.16304334,
       -0.20679239, -0.40316623, -0.53431714, -0.58094382, -0.60935612,
       -0.65591264, -0.80088411, -0.83391625, -0.88069878, -1.00236445,
       -1.40870898, -1.41756227, -1.43311447, -1.45275421, -1.62997497,
       -1.67675751, -1.77949612, -1.97635662, -2.00405624, -2.0229833 ,
       -2.07909523, -2.1668042 , -2.21609764, -2.26424392, -2.27595573,
       -2.37869434, -2.47243437, -2.54750862, -2.57555484, -2.65951766,
       -2.67829345, -2.83189156, -2.87515395, -5.5324972 ])
```

[2309]:
```
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.show()
```



[2310]:
```
log_reg = LogisticRegression(C=300, penalty='l1', solver='liblinear')
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_val)
```

[2311]:
```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_val, y_pred)
```

[2311]:
```
array([[90, 18],
       [19, 52]])
```

[2312]:
```
from sklearn.metrics import f1_score
print(f1_score(y_val, y_pred))
```

```
0.7375886524822696
```

[2313]:
```
y_pred_test = log_reg.predict(X_submission)
```

[2314]:
```
y_pred_test.shape
```

[2314]:
```
(418,)
```

[2315]:
```
submission_data = np.c_[titanic_test_df["PassengerId"].values, y_pred_test]
submission_df = pd.DataFrame(data = submission_data, columns = ["PassengerId",␣
↪"Survived"])
submission_df['PassengerId'] = submission_df['PassengerId'].astype('int64')
```

```
submission_df['Survived'] = submission_df['Survived'].astype('int64')
submission_df
```

[2315]:

```
     PassengerId  Survived
0            892         0
1            893         0
2            894         0
3            895         0
4            896         1
..           ...       ...
413         1305         0
414         1306         1
415         1307         0
416         1308         0
417         1309         0

[418 rows x 2 columns]
```

[2316]:

```
submission_df.to_csv("/Users/anaswarjayakumar/Downloads/titanic_log_reg.csv",␣
↪index = False)
```

9.2 Naive Bayes Classification

[2317]:

```
# import titanic data
titanic_train_df_copy = titanic_train_df.copy()
titanic_test_df_copy = titanic_test_df.copy()

titanic_train_df_copy.drop(columns = ['Name', 'Cabin', 'Ticket'], inplace =␣
↪True)
titanic_test_df_copy.drop(columns = ['Name', 'Cabin', 'Ticket'], inplace = True)
```

[2318]:

```
titanic_train_df_copy.head()
```

[2318]:

```
   PassengerId  Survived  Pclass  Sex      Age  SibSp  Parch     Fare  Embarked  \
0            1         0       3    1  22.0000      1      0   7.2500         3
1            2         1       1    0  38.0000      1      0  71.2833         0
2            3         1       3    0  26.0000      0      0   7.9250         3
3            4         1       1    0  35.0000      1      0  53.1000         3
4            5         0       3    1  35.0000      0      0   8.0500         3

       FPP
0   3.6250
1  35.6416
2   7.9250
3  26.5500
4   8.0500
```

[2319]:
```
titanic_test_df_copy.head()
```

[2319]:
```
   PassengerId  Pclass  Sex      Age  SibSp  Parch     Fare  Embarked
0          892       3    1  34.5000      0      0   7.8292         2
1          893       3    0  47.0000      1      0   7.0000         3
2          894       2    1  62.0000      0      0   9.6875         2
3          895       3    1  27.0000      0      0   8.6625         3
4          896       3    0  22.0000      1      1  12.2875         3
```

[2320]:
```
def convert_age(df):
    df["age_category"] = 0
    for i, row in df.iterrows():
        age = row['Age']
        age_category = 0
        if age <= 5:
            age_category = 1
        elif 5 < age <= 10:
            age_category = 2
        elif 10 < age <= 15:
            age_category = 3
        elif 15 < age <= 20:
            age_category = 4
        elif 20 < age <= 25:
            age_category = 5
        elif 25 < age <= 30:
            age_category = 6
        elif 30 < age <= 40:
            age_category = 7
        elif 40 < age <= 50:
            age_category = 8
        elif 50 < age <= 60:
             age_category = 9
        else:
            age_category = 10

        df.at[i,"age_category"] = age_category
```

[2321]:
```
convert_age(titanic_train_df_copy)
titanic_train_df_copy
```

[2321]:
```
     PassengerId  Survived  Pclass  Sex      Age  SibSp  Parch     Fare  \
0              1         0       3    1  22.0000      1      0   7.2500
1              2         1       1    0  38.0000      1      0  71.2833
2              3         1       3    0  26.0000      0      0   7.9250
3              4         1       1    0  35.0000      1      0  53.1000
4              5         0       3    1  35.0000      0      0   8.0500
..           …         …       …  …        …      …      …        …
```

```
886          887         0        2    1 27.0000      0       0 13.0000
887          888         1        1    0 19.0000      0       0 30.0000
888          889         0        3    0 27.9157      1       2 23.4500
889          890         1        1    1 26.0000      0       0 30.0000
890          891         0        3    1 32.0000      0       0  7.7500

     Embarked      FPP  age_category
0           3   3.6250             5
1           0  35.6416             7
2           3   7.9250             6
3           3  26.5500             7
4           3   8.0500             7
..        ...      ...           ...
886         3  13.0000             6
887         3  30.0000             4
888         3   5.8625             6
889         0  30.0000             6
890         2   7.7500             7

[891 rows x 11 columns]
```

[2322]:
```
convert_age(titanic_test_df_copy)
titanic_test_df_copy
```

[2322]:
```
     PassengerId  Pclass  Sex      Age  SibSp  Parch      Fare  Embarked  \
0            892       3    1  34.5000      0      0    7.8292         2
1            893       3    0  47.0000      1      0    7.0000         3
2            894       2    1  62.0000      0      0    9.6875         2
3            895       3    1  27.0000      0      0    8.6625         3
4            896       3    0  22.0000      1      1   12.2875         3
..           ...     ...  ...      ...    ...    ...       ...       ...
413         1305       3    1  30.2727      0      0    8.0500         3
414         1306       1    0  39.0000      0      0  108.9000         0
415         1307       3    1  38.5000      0      0    7.2500         3
416         1308       3    1  30.2727      0      0    8.0500         3
417         1309       3    1  30.2727      1      1   22.3583         0

     age_category
0               7
1               8
2              10
3               6
4               5
..            ...
413             7
414             7
415             7
```

```
416             7
417             7

[418 rows x 9 columns]
```

[2323]:
```
def convert_fare(df):
    df["fare_category"] = 0
    for i, row in df.iterrows():
        fare = row['Fare']
        fare_category = 0
        if fare <= 5:
            fare_category = 1
        elif 5 < fare <= 15:
            fare_category = 2
        elif 15 < fare <= 25:
            fare_category = 3
        elif 25 < fare <= 35:
            fare_category = 4
        elif 35 < fare <= 45:
            fare_category = 5
        elif 45 < fare <= 55:
            fare_category = 6
        elif 55 < fare <= 65:
            fare_category = 7
        elif 65 < fare <= 100:
            fare_category = 8
        else:
            fare_category = 9

        df.at[i,"fare_category"] = fare_category
```

[2324]:
```
convert_fare(titanic_train_df_copy)
titanic_train_df_copy
```

[2324]:
```
     PassengerId  Survived  Pclass  Sex      Age  SibSp  Parch     Fare  \
0              1         0       3    1  22.0000      1      0   7.2500
1              2         1       1    0  38.0000      1      0  71.2833
2              3         1       3    0  26.0000      0      0   7.9250
3              4         1       1    0  35.0000      1      0  53.1000
4              5         0       3    1  35.0000      0      0   8.0500
..           …         …       …  …      …     …      …
886          887         0       2    1  27.0000      0      0  13.0000
887          888         1       1    0  19.0000      0      0  30.0000
888          889         0       3    0  27.9157      1      2  23.4500
889          890         1       1    1  26.0000      0      0  30.0000
890          891         0       3    1  32.0000      0      0   7.7500
```

```
     Embarked      FPP  age_category  fare_category
0           3   3.6250             5              2
1           0  35.6416             7              8
2           3   7.9250             6              2
3           3  26.5500             7              6
4           3   8.0500             7              2
..        …        …             …              …
886         3  13.0000             6              2
887         3  30.0000             4              4
888         3   5.8625             6              3
889         0  30.0000             6              4
890         2   7.7500             7              2

[891 rows x 12 columns]
```

[2325]:
```
convert_fare(titanic_test_df_copy)
titanic_test_df_copy
```

[2325]:
```
     PassengerId  Pclass  Sex      Age  SibSp  Parch      Fare  Embarked  \
0            892       3    1  34.5000      0      0    7.8292         2
1            893       3    0  47.0000      1      0    7.0000         3
2            894       2    1  62.0000      0      0    9.6875         2
3            895       3    1  27.0000      0      0    8.6625         3
4            896       3    0  22.0000      1      1   12.2875         3
..           …     …   …        …      …      …         …
413         1305       3    1  30.2727      0      0    8.0500         3
414         1306       1    0  39.0000      0      0  108.9000         0
415         1307       3    1  38.5000      0      0    7.2500         3
416         1308       3    1  30.2727      0      0    8.0500         3
417         1309       3    1  30.2727      1      1   22.3583         0

     age_category  fare_category
0               7              2
1               8              2
2              10              2
3               6              2
4               5              2
..            …              …
413             7              2
414             7              9
415             7              2
416             7              2
417             7              3

[418 rows x 10 columns]
```

[2326]:
```python
def convert_parch(df):
    df["parch_category"] = 0
    for i, row in df.iterrows():
        parch = row['Parch']
        parch_category = 0
        if parch <= 0:
            parch_category = 1
        elif 0 < parch <= 1:
            parch_category = 2
        elif 1 < parch <= 2:
            parch_category = 3
        elif 2 < parch <= 3:
            parch_category = 4
        elif 3 < parch <= 5:
            parch_category = 5
        else:
            parch_category = 6

        df.at[i,"parch_category"] = parch_category
```

[2327]:
```python
convert_parch(titanic_train_df_copy)
titanic_train_df_copy
```

[2327]:
```
     PassengerId  Survived  Pclass  Sex      Age  SibSp  Parch     Fare  \
0              1         0       3    1  22.0000      1      0   7.2500
1              2         1       1    0  38.0000      1      0  71.2833
2              3         1       3    0  26.0000      0      0   7.9250
3              4         1       1    0  35.0000      1      0  53.1000
4              5         0       3    1  35.0000      0      0   8.0500
..           …         …       …   …        …      …      …
886          887         0       2    1  27.0000      0      0  13.0000
887          888         1       1    0  19.0000      0      0  30.0000
888          889         0       3    0  27.9157      1      2  23.4500
889          890         1       1    1  26.0000      0      0  30.0000
890          891         0       3    1  32.0000      0      0   7.7500

     Embarked      FPP  age_category  fare_category  parch_category
0           3   3.6250             5              2               1
1           0  35.6416             7              8               1
2           3   7.9250             6              2               1
3           3  26.5500             7              6               1
4           3   8.0500             7              2               1
..        …        …            …             …              …
886         3  13.0000             6              2               1
887         3  30.0000             4              4               1
888         3   5.8625             6              3               3
889         0  30.0000             6              4               1
```

```
890          2  7.7500              7             2                1

[891 rows x 13 columns]
```

[2328]:
```
convert_parch(titanic_test_df_copy)
titanic_test_df_copy
```

[2328]:
```
     PassengerId  Pclass  Sex      Age  SibSp  Parch      Fare  Embarked  \
0            892       3    1  34.5000      0      0    7.8292         2
1            893       3    0  47.0000      1      0    7.0000         3
2            894       2    1  62.0000      0      0    9.6875         2
3            895       3    1  27.0000      0      0    8.6625         3
4            896       3    0  22.0000      1      1   12.2875         3
..           …      …   …      …      …    …      …       …
413         1305       3    1  30.2727      0      0    8.0500         3
414         1306       1    0  39.0000      0      0  108.9000         0
415         1307       3    1  38.5000      0      0    7.2500         3
416         1308       3    1  30.2727      0      0    8.0500         3
417         1309       3    1  30.2727      1      1   22.3583         0

     age_category  fare_category  parch_category
0               7              2               1
1               8              2               1
2              10              2               1
3               6              2               1
4               5              2               2
..            …             …              …
413             7              2               1
414             7              9               1
415             7              2               1
416             7              2               1
417             7              3               2

[418 rows x 11 columns]
```

[2329]:
```
# cols_to_drop = ['Fare', 'Age', 'Parch']
# titanic_train_df_copy.drop(columns = cols_to_drop, inplace = True)
# titanic_test_df_copy.drop(columns = cols_to_drop, inplace = True)
```

[2330]:
```
# Take the survived column and set it as the target column since the loss␣
↪variable is what is being predicted
target = ['Survived']
# Create the list of predictors variables
predictors = ['Pclass', 'Sex', 'SibSp', 'Embarked', 'age_category',␣
↪'fare_category', 'parch_category']
```

[2331]:
```
for col in predictors:
    print(col)
    print(titanic_train_df_copy[col].unique())
    print(titanic_test_df_copy[col].unique())
```

```
Pclass
[3 1 2]
[3 2 1]
Sex
[1 0]
[1 0]
SibSp
[1 0 3 4 2 5 8]
[0 1 2 3 4 5 8]
Embarked
[3 0 2 1]
[2 3 0]
age_category
[ 5  7  6  9  1  3  4  2 10  8]
[ 7  8 10  6  5  3  4  9  2  1]
fare_category
[2 8 6 3 4 5 9 7 1]
[2 4 3 8 7 1 9 6 5]
parch_category
[1 2 3 5 4 6]
[1 2 4 3 5 6]
```

[2332]:
```
X = titanic_train_df_copy[predictors].values
y = titanic_train_df_copy[target].values

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.20)
```

[2333]:
```
from sklearn.naive_bayes import CategoricalNB
clf = CategoricalNB(class_prior=[0.72, 0.28], fit_prior = False)
clf.fit(X_train, y_train)
```

[2333]:
```
CategoricalNB(class_prior=[0.72, 0.28], fit_prior=False)
```

[2334]:
```
print(X_train.shape)
print(X_val.shape)
```

```
(712, 7)
(179, 7)
```

[2335]:
```
y_pred_nb = clf.predict(X_train)
confusion_matrix(y_train, y_pred_nb)
```

[2335]:
```
array([[383,  56],
       [111, 162]])
```

[2336]:
```
y_pred_nb = clf.predict(X_val)
confusion_matrix(y_val, y_pred_nb)
```

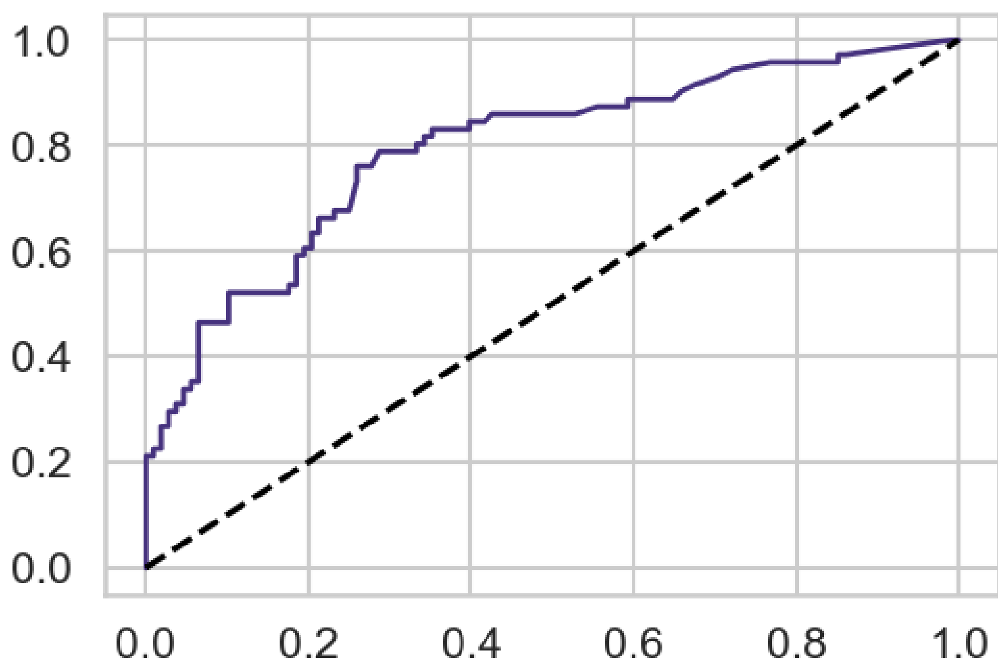
[2336]:
```
array([[99, 11],
       [32, 37]])
```

[2337]:
```
from sklearn.metrics import f1_score
print(f1_score(y_val, y_pred_nb))
```

```
0.6324786324786326
```

[2282]:
```
y_prob = clf.predict_proba(X_val)

fpr, tpr, thresholds = roc_curve(y_val, y_prob[:, 1])
roc_auc = auc(fpr, tpr)
print(roc_auc)

plt.plot(fpr, tpr, linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.show()
```

```
0.79349243609807
```



[2243]:
```
X_submission = titanic_test_df_copy[predictors].values
X_submission.shape
```

[2243]: (418, 7)

[2244]:
```
X_train.shape
```

[2244]: (712, 7)

[2245]:
```
y_pred_submission = clf.predict(X_submission)
```

[2246]:
```
submission_data = np.c_[titanic_test_df["PassengerId"].values,␣
 ↪y_pred_submission]
submission_df = pd.DataFrame(data = submission_data, columns = ["PassengerId",␣
 ↪"Survived"])
submission_df['PassengerId'] = submission_df['PassengerId'].astype('int64')
submission_df['Survived'] = submission_df['Survived'].astype('int64')
submission_df
```

[2246]:
```
     PassengerId  Survived
0            892         0
1            893         0
2            894         0
3            895         0
4            896         1
..           …        …
413         1305         0
414         1306         1
415         1307         0
416         1308         0
417         1309         0

[418 rows x 2 columns]
```

[2094]:
```
submission_df.to_csv("/Users/anaswarjayakumar/Downloads/titanic_naive_bayes.
 ↪csv", index = False)
```

9.3 XGBoost Classifier

Note - XGBoost Classifier uses the features generated by the Logistic Regression, so the Logistic Regression needs to be run first before running the XGBoost Classifer. Specifically dont run the Naive Bayes immediately after running the Logistic Regression

[2293]:
```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, KFold

params = {
```

```
    'eta': 0.3,
    'max_depth': 10
}
xgb_model = XGBClassifier()
xgb_model.set_params(**params)
xgb_model.fit(X_train, y_train)

# Cross Validation
scores = cross_val_score(xgb_model, X_train, y_train, cv = 5)
print("Mean Cross Validation score: ", scores.mean())

# KFold
k_fold = KFold(n_splits = 10, shuffle = True)
kf_cv_scores = cross_val_score(xgb_model, X_train, y_train, cv = k_fold)
print("KFold Cross Validation score: ", kf_cv_scores.mean())
```

```
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
Mean Cross Validation score:  0.7795232936078007
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
```

```
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:53] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
eval_metric if you'd like to restore the old behavior.
[17:06:54] WARNING: /Users/anaswarjayakumar/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.8/xgboost/src/learner.cc:1094: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set
```

```
eval_metric if you'd like to restore the old behavior.
KFold Cross Validation score:  0.7907472613458528
```

[2294]:
```python
y_pred_xgb = xgb_model.predict(X_val)
confusion_matrix(y_val, y_pred_xgb)
```

[2294]:
```
array([[91, 17],
       [19, 52]])
```

[2295]:
```python
predictions = [round(value) for value in y_pred_xgb]
accuracy = accuracy_score(y_val, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 79.89%
```

[2296]:
```python
y_pred_test = log_reg.predict(X_submission)
```

[2297]:
```python
submission_data = np.c_[titanic_test_df["PassengerId"].values, y_pred_test]
submission_df = pd.DataFrame(data = submission_data, columns = ["PassengerId",␣
 ↪"Survived"])
submission_df['PassengerId'] = submission_df['PassengerId'].astype('int64')
submission_df['Survived'] = submission_df['Survived'].astype('int64')
submission_df
```

[2297]:
```
     PassengerId  Survived
0            892         0
1            893         0
2            894         0
3            895         0
4            896         1
..           …         …
413         1305         0
414         1306         1
415         1307         0
416         1308         0
417         1309         0

[418 rows x 2 columns]
```

[2132]:
```python
submission_df.to_csv("/Users/anaswarjayakumar/Downloads/titanic_xgboost.csv",␣
 ↪index = False)
```

10 Conclusion

10.1 Data preparation, exploration, visualization

Some of the data preparation techniques that were used in Assignment 2 such as Label Encoding were carried over to Assignment 3. Label Encoding was performed for both the training and test data. In addition, arrays for the features and the response variable were created as well.

Specifically, I set aside the Fare, Age, Parch, SibSp, Embarked, Pclass, and Sex variables as my predictor variables and set aside the Survived variable as the response variable since the goal is to predict which passengers survived and which passengers didn't. For the predictor variables, I arrived at Fare, Age, Parch, SibSp, Embarked, Pclass, and Sex variables based on the visualizations that I did. I dropped certain columns such as name, ticket, and cabin as I did not think that these columns would be helpful when predicting which passengers survived and which passengers didn't. However, certain columns that I didn't think would be would useful otherwise, ended up being useful in predicting which passengers did and did not survive.

To better visualize the data, I created several plots and graphs for both the categorical and numerical features that were present in the data, some of which were from Kaggle and some my own. Sex, Class, Family Count, and Embarked were classified as categorical features whereas Age, and Fare were considered as numerical features. In addition, I also created box plots as well as KDE plots. The plots and graphs allowed me to gain further insight into the likelihood of a passenger surviving based on certain parameters as well as the respective distribution. From the multivariate box plots, it is easy to see that the passengers in second class can easily be separated into survived and not survived based on their age. From the bar plots, it is easy to see how certain attributes such as age, sex, class, fare, and family count affect a passenger's likelihood of survival, while the histograms do aid in understanding the overall distribution of the data. In addition, the KDE plots were useful in better understanding the PDF of the continuous features that were present in the data. Lastly the correlation matrix was also useful in depicting the correlation between the different features and this is quite useful especially when selecting which features to use in the Logistic Regression, Naive Bayes, and the XGBoost classifier.

10.2 Research Design/Review results, evaluate models

In this assignment, three methods were used: Logistic Regression, Naive Bayes, and XGBoost. The implementation of the Logistic Regression method via GridSearchCV generated a score of 0.8005811090318133, a ROC-AUC of 0.8319640062597811, and and F1 score of 0.7375886524822696. However, the Naive Bayes method generated a F1 score of 0.6324786324786326 and a ROC-AUC score of 0.79349243609807. As the XGBoost generated much better results in Assignment 2, I then tried the XGBoost classier to see if my results would be any better. However, the accuracy was only 79.89%. When implemented via GridSearchCV, I do think that the Logistic Regression performs slightly better than the Naive Bayes and the XGBoost classifier solely based on the aforementioned scores. I definitely think that the scores generated by Logistic Regression, Naive Bayes and XGBoost models could be improved had I been more strategic about which features to train the model on and I did notice that after several Kaggle submissions my score ended up being the same.

10.3 Exposition and Management Recommendations

The results in Kaggle could definitely be improved which means there is still more work to be done in terms of feature generation. Many of the kagglers reported much better scores and one way to improve my scores is by considering combinations of different features to help improve the results. Some of the Kagglers were able to extract additional information out of certain columns. For example, one kaggler was able to extract information such as the title of the passenger (Mr. Mrs. Ms.) from the name column. Out of the two modeling methods that were used, I would recommend the Logistic Regression method as it is easier to implement and is much more robust compared to the Naive Bayes. When implementing the Naive Bayes method, there were certain values that were exclusive to the test set and therefore, the algorithm for the Naive Bayes ran into issues.