In [151]:
```python
# Basic Libraries
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt # we only need pyplot
from plotnine import *
sb.set() # set the default Seaborn style for graphics
from category_encoders.woe import WOEEncoder
from category_encoders.leave_one_out import LeaveOneOutEncoder
from category_encoders.ordinal import OrdinalEncoder
```

Titanic Dataset

In [4]:
```python
titanic_data=pd.read_csv('train.csv')
```

In [5]:
```python
print('Dimensions of training data: ',titanic_data.shape)
titanic_data.head(3)
```

```
Dimensions of training data:  (891, 12)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	

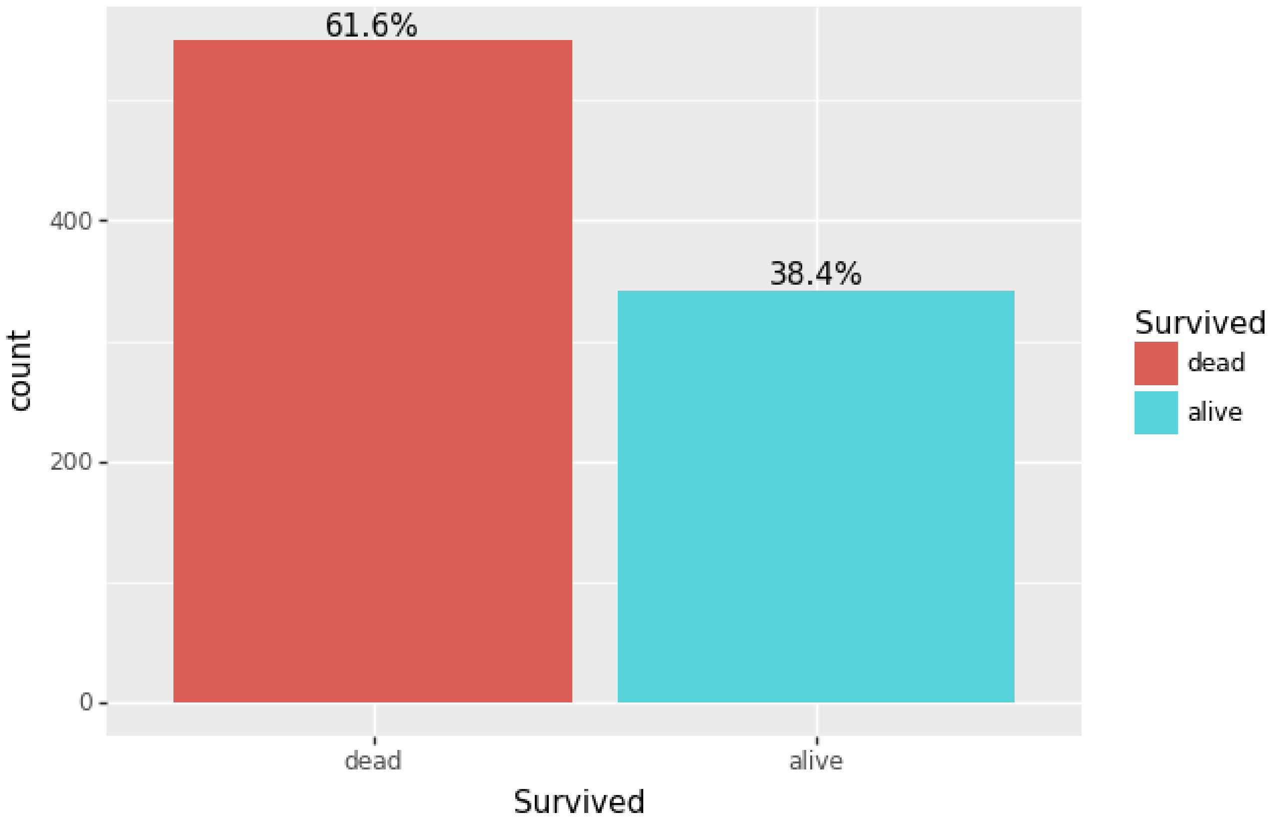
- Summary:
  - 10 predictors/features
  - 1 response variable
  - 1 ID column

Response Variable: Survived

In [6]:
```python
# convert the response categorical
response_copy=titanic_data['Survived'].copy()
titanic_data['Survived']=pd.Categorical(titanic_data['Survived'])

# rename response categories to more meaningful ones
titanic_data['Survived']=titanic_data['Survived'].cat.rename_categories({0:'dead',1:'al
```

In [7]:
```
ggplot(titanic_data,aes(x='Survived',fill='Survived'))+geom_bar()+geom_text(
    aes(label='stat(prop)*100', group=1),
     stat='count',
     nudge_y=0.125,
     va='bottom',
     format_string='{:.1f}%'
 )
```



Out[7]: <ggplot: (-9223371963729912084)>

- Observation:
  - slightly imbalanced

Basic Exploratory data analysis

In [8]:
```
titanic_data.dtypes
```

Out[8]:
```
PassengerId       int64
Survived       category
Pclass            int64
Name             object
Sex              object
Age             float64
SibSp             int64
Parch             int64
Ticket           object
Fare            float64
Cabin            object
Embarked         object
dtype: object
```

In [9]:
```
print("Number of levels:")
print()
for var in titanic_data.columns:
    print("{:<20}:{}".format(var,len(titanic_data[var].unique())))
```

```
Number of levels:

PassengerId         :891
Survived            :2
Pclass              :3
Name                :891
Sex                 :2
Age                 :89
SibSp               :7
Parch               :7
Ticket              :681
Fare                :248
Cabin               :148
Embarked            :4
```

- Obersvation:
  - all names are unique, though i won't assume it's completely useless
  - similarly, there are a **lot of levels for Ticket and Cabin variable**, but it may hold some patterens and information

dealing with the NAs

In [10]:
```
titanic_data.isna().sum()
```

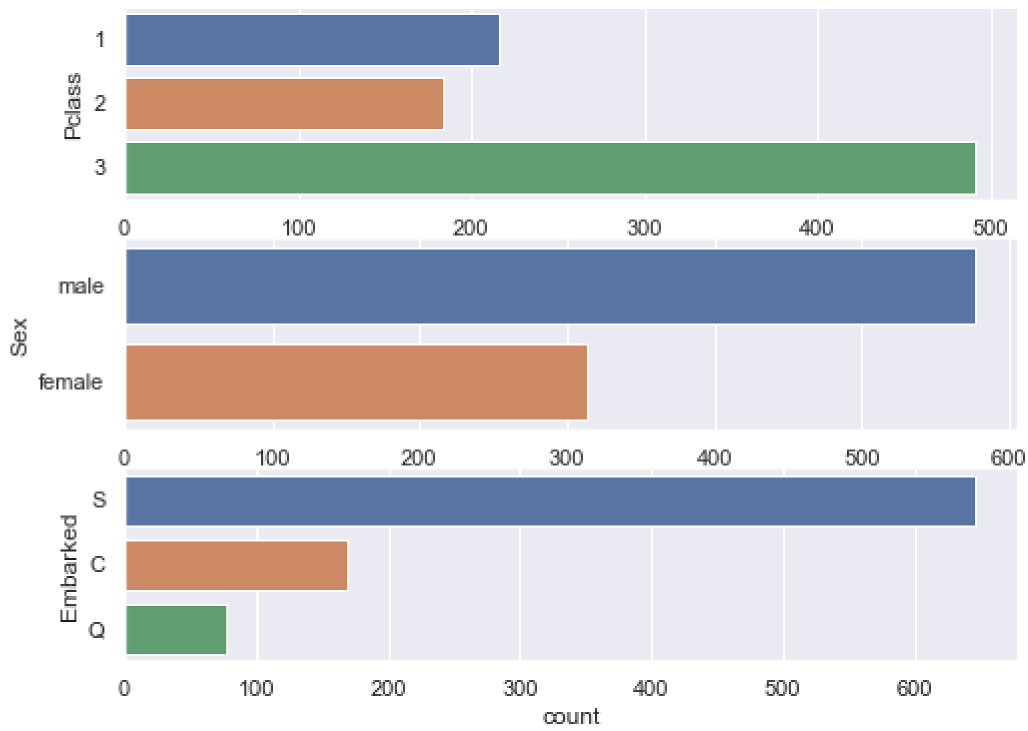
Out[10]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [11]:
```
    # fill Age variable with random integers between mean and max
from random import randint
titanic_data['Age'].fillna(randint(int(titanic_data['Age'].min()),titanic_data['Age'].m
    # fill Cabin NAs with 'Missing' because there are too many
titanic_data['Cabin'].fillna('Missing',inplace=True)
    # fill embarked NAs with its mode
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],inplace=True)
```

- Steps:
  - dealt with NAs temporarily to make visualizations
- Observation:
  - there are newborns who's Age's are in decimals and less than 1

Visualizing Pclass, Sex, Embarked

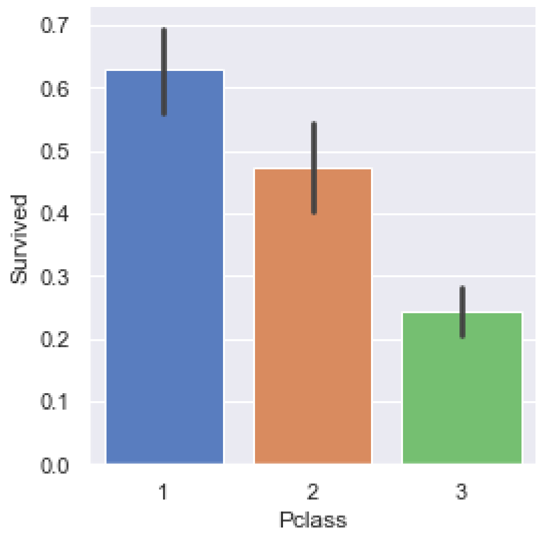
In [12]:
```
# uni-variate visuals
f, axes = plt.subplots(3,1 , figsize=(8, 6))
for i,var in enumerate(['Pclass','Sex','Embarked']):
    sb.countplot(data=titanic_data,y=var,ax=axes[i])
```



- Observation:
  - most passengers are from Pclass 3, are male and came from Southhampton

In [13]:
```
# convert to contegoric
for var in ['Pclass','Sex','Embarked']:
    titanic_data[var]=pd.Categorical(titanic_data[var])
```

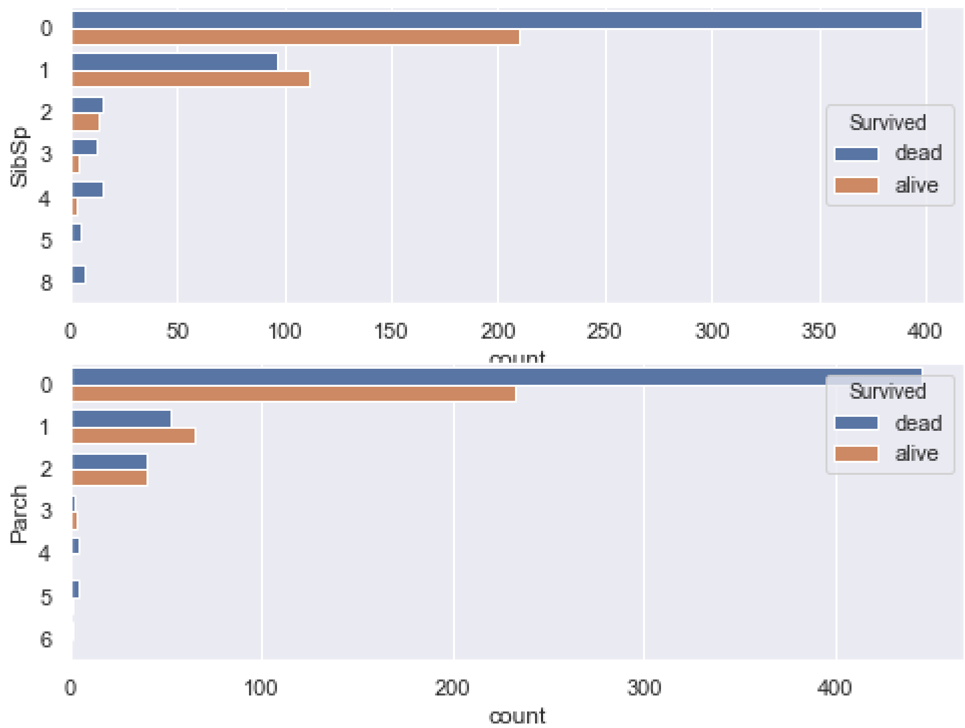
In [14]:
```
temp_data=pd.concat([titanic_data[['Pclass','Sex','Embarked']],response_copy],axis=1)
for i,var in enumerate(['Pclass','Sex','Embarked']):
    sb.catplot(x=var,y='Survived',data=temp_data,kind='bar',palette='muted',height=4, a
```



- Observations:
  - people from class 3, males and from Southampton are more likely to die
  - people from class 1 and females are more likely to live

Visualising SibSp and Parch

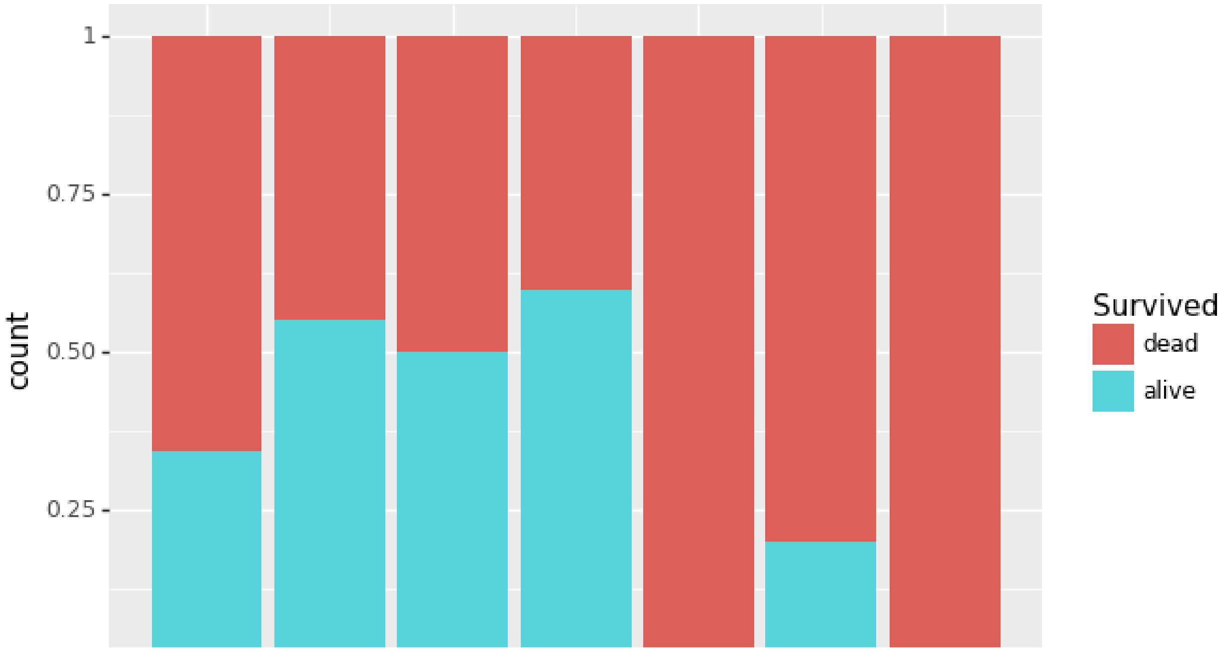
In [15]:
```
f, axes = plt.subplots(2,1 , figsize=(8, 6))
for i,var in enumerate(['SibSp','Parch']):
    sb.countplot(data=titanic_data,y=var,hue='Survived',ax=axes[i])
```



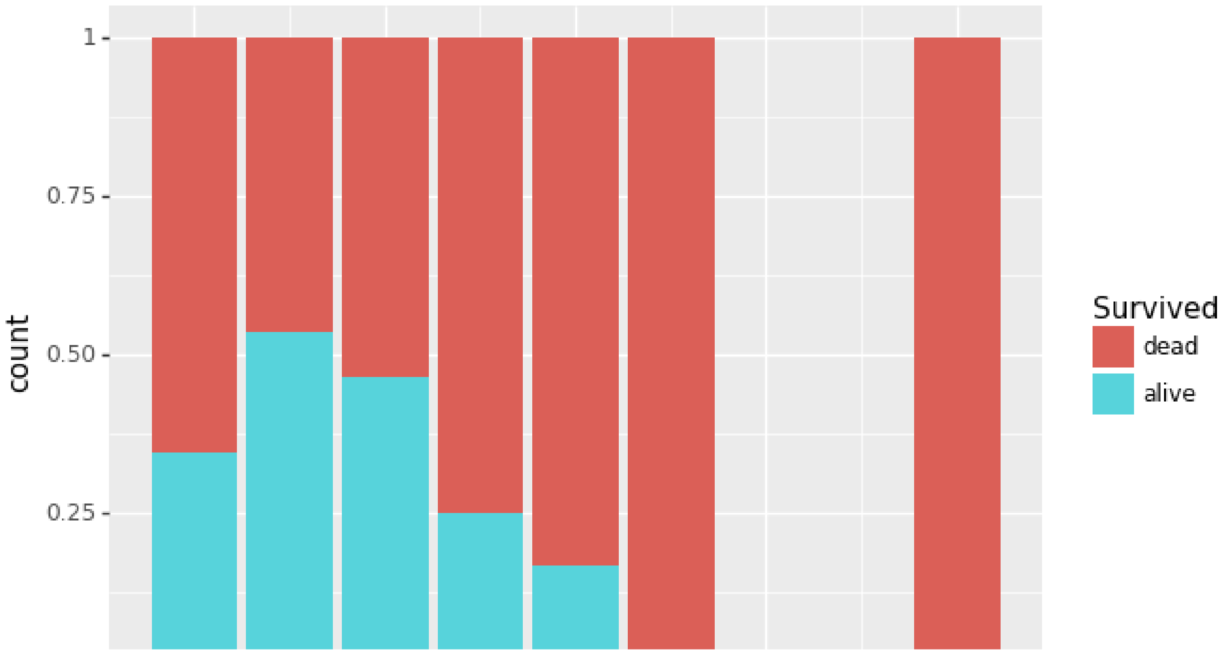
- Observation:

- most passengers have around 1-2 family members

In [16]:
```
ggplot(titanic_data,aes(x='Parch',fill='Survived'))+geom_bar(position='fill')
```



In [17]:
```
ggplot(titanic_data,aes(x='SibSp',fill='Survived'))+geom_bar(position='fill')
```



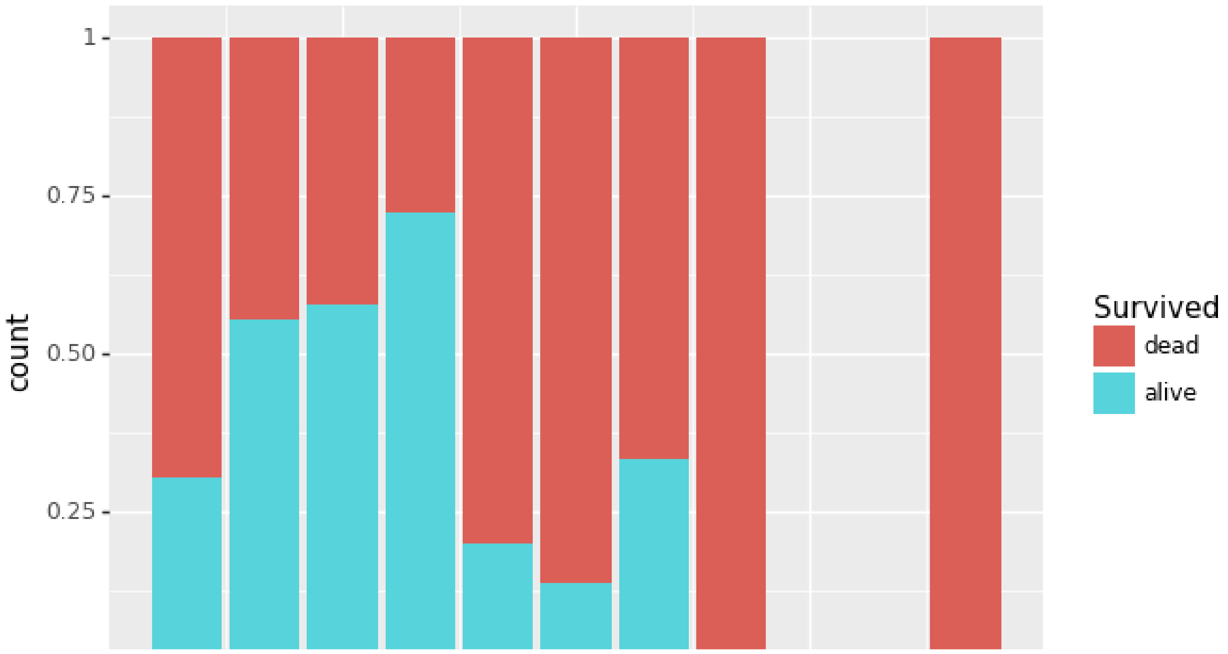
- Observation:
    - as the number of family members increase the likelihood of survival decrease

new feature: family_size

- combination of relevant features usually lead to better predictors

In [18]:
```
# new feature: sum of SibSp and Parch
titanic_data['family_size']=titanic_data['SibSp']+titanic_data['Parch']+1
```

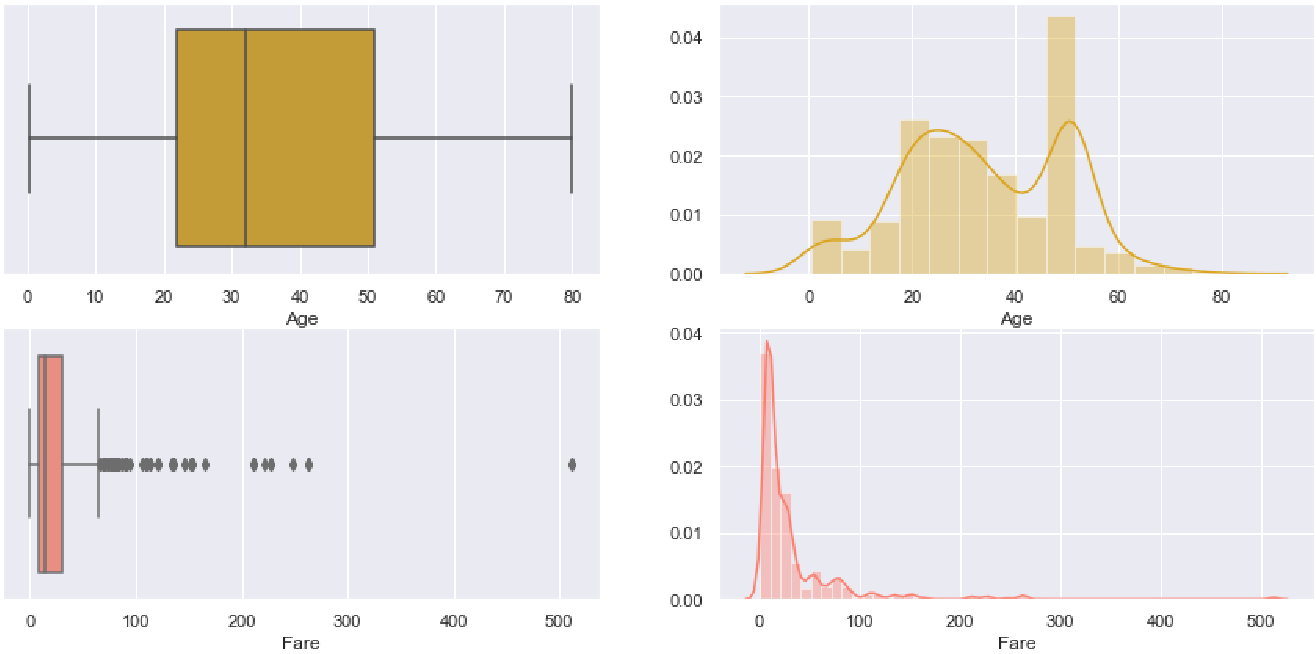
In [19]:
```
ggplot(titanic_data,aes(x='family_size',fill='Survived'))+geom_bar(position='fill')
```



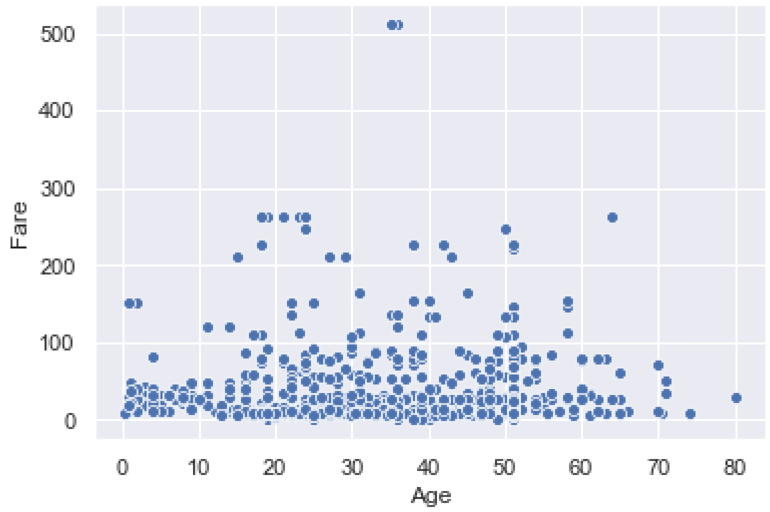
- Observation:
  - smaller families a more likely to survive than larger ones
  - however, being alone indicates smaller chance of survival

Visualising Age and Fare

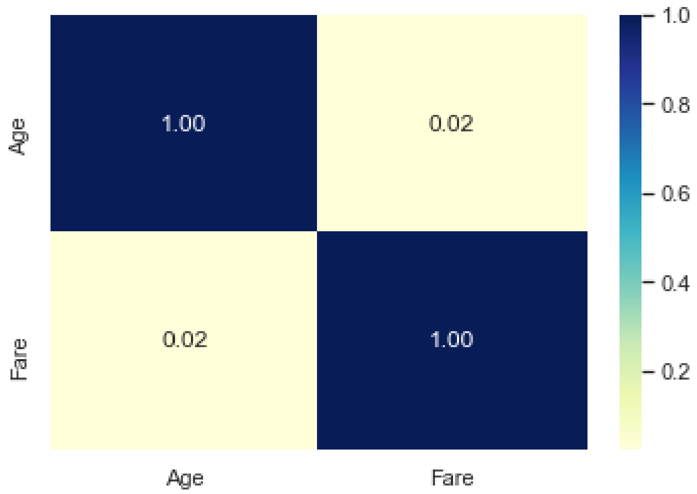
In [20]:
```
f, axes = plt.subplots(2,2 , figsize=(15, 7))
color_list=['goldenrod','salmon']
for i,var in enumerate(['Age','Fare']):
    sb.boxplot(titanic_data[var],ax=axes[i,0],color=color_list[i])
    sb.distplot(titanic_data[var],ax=axes[i,1],color=color_list[i])
```



In [21]:
```
sb.scatterplot(x='Age', y='Fare',data=titanic_data)
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1106ccac308>

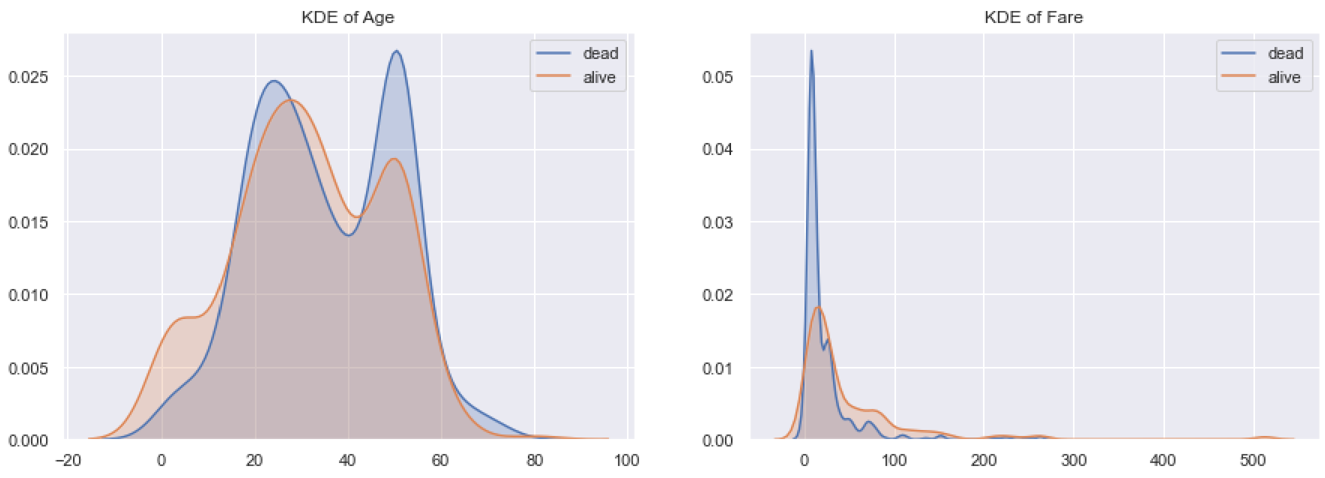


In [22]:
```
sb.heatmap(titanic_data[['Age','Fare']].corr(),annot=True,cmap="YlGnBu",fmt='.2f')
```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x1106d051588>



- Observation:
  - there are very suspicious looking fares that are much more than usual fares

In [23]:
```
f, axes = plt.subplots(1,2 , figsize=(15, 5))
for i,var in enumerate(['Age','Fare']):
    for level in titanic_data['Survived'].unique():
        sb.kdeplot(titanic_data[titanic_data['Survived']==level][var],shade=True,label=
```



- Obersvation:
    - children under the age of 10 are more likely to survive
    - people who paid more for their fare are more likely to survive

Visualising Cabin variable

In [24]:
```python
titanic_data['Cabin'].unique()
```

Out[24]:
```
array(['Missing', 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [25]:
```python
def multiple_cabins(target):
    if target=='Missing':
        return False
    elif len(target)>4:
        return True
    else:
        return False
```

In [26]:
```python
titanic_data[titanic_data['Cabin'].apply(lambda x: multiple_cabins(x))].sort_values(['C
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cab
872	873	dead	1	Carlsson, Mr. Frans Olof	male	33.00	0	0	695	5.0000	B B B
679	680	alive	1	Cardeza, Mr. Thomas Drake Martinez	male	36.00	0	1	PC 17755	512.3292	B B B
742	743	alive	1	Ryerson, Miss. Susan Parker "Suzette"	female	21.00	2	2	PC 17608	262.3750	B B B B
311	312	alive	1	Ryerson, Miss. Emily Borie	female	18.00	2	2	PC 17608	262.3750	B B B B
				Baxter, Mr.					PC		B

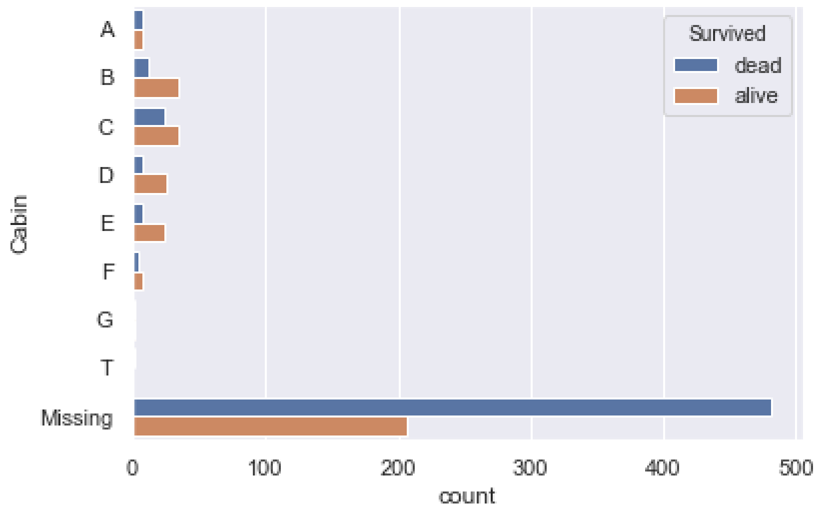
- it is clear that when a passenger is assigned multiple cabins, their fares are usually outliers
- it's impossible to determine the right way to find their actual fares but i will divide their fares by the number of cabins to bring the fares closer to their actual value

In [27]:
```python
titanic_data['Cabin']=titanic_data['Cabin'].apply(lambda x: x if x=='Missing' else x[0]
```

- Cabin Variable:
  - reduced its cardinality by grouping alphabets together
  - levels have 2 alphebets; will just take the first occuring one

In [28]:
```
sb.countplot(data=titanic_data,y='Cabin',hue='Survived',order=['A','B','C','D','E','F',
```

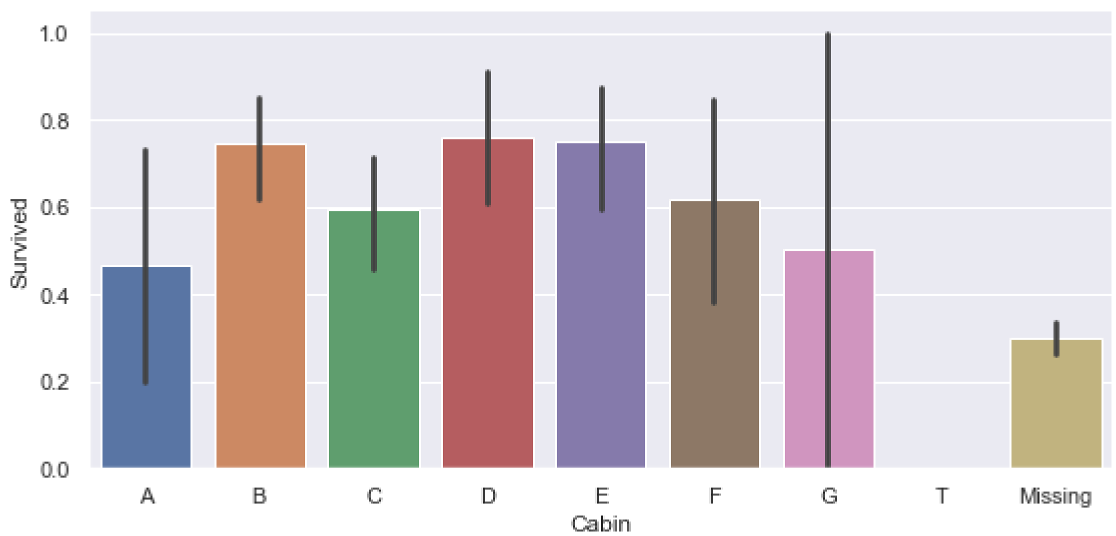
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1106d207cc8>



- Observation:
  - some levels are quite rare

In [29]:
```
temp_data=pd.concat([titanic_data['Cabin'],response_copy],axis=1)
sb.catplot(x='Cabin',y='Survived',data=temp_data,kind='bar',height=4, aspect=2,order=['
```

Out[29]: <seaborn.axisgrid.FacetGrid at 0x1106b7699c8>



- Observation:
  - people with a missing cabin assigned are more likely to die

Visualising Ticket variable

In [30]:
```python
titanic_data['Ticket'].unique()
```

Out[30]:
```
array(['A/5 21171', 'PC 17599', 'STON/O2. 3101282', '113803', '373450',
       '330877', '17463', '349909', '347742', '237736', 'PP 9549',
       '113783', 'A/5. 2151', '347082', '350406', '248706', '382652',
       '244373', '345763', '2649', '239865', '248698', '330923', '113788',
       '347077', '2631', '19950', '330959', '349216', 'PC 17601',
       'PC 17569', '335677', 'C.A. 24579', 'PC 17604', '113789', '2677',
       'A./5. 2152', '345764', '2651', '7546', '11668', '349253',
       'SC/Paris 2123', '330958', 'S.C./A.4. 23567', '370371', '14311',
       '2662', '349237', '3101295', 'A/4. 39886', 'PC 17572', '2926',
       '113509', '19947', 'C.A. 31026', '2697', 'C.A. 34651', 'CA 2144',
       '2669', '113572', '36973', '347088', 'PC 17605', '2661',
       'C.A. 29395', 'S.P. 3464', '3101281', '315151', 'C.A. 33111',
       'S.O.C. 14879', '2680', '1601', '348123', '349208', '374746',
       '248738', '364516', '345767', '345779', '330932', '113059',
       'SO/C 14885', '3101278', 'W./C. 6608', 'SOTON/OQ 392086', '343275',
       '343276', '347466', 'W.E.P. 5734', 'C.A. 2315', '364500', '374910',
       'PC 17754', 'PC 17759', '231919', '244367', '349245', '349215',
       '35281', '7540', '3101276', '349207', '343120', '312991', '349249',
       '371110', '110465', '2665', '324669', '4136', '2627',
```

In [31]:
```python
def letters_present(target):
    for i in target:
        if i.isalpha():
            return True
    return False

def extract_letters(target):
    result=''
    for i in target:
        if i.isalpha():
            result+=i
    return result
```

In [32]:
```python
titanic_data['ticket_letters']=titanic_data['Ticket'].apply(lambda x: 'present' if lett
```

```
In [33]: ggplot(titanic_data,aes(x='ticket_letters',fill='Survived'))+geom_bar(position='fill')
```

- Observation:
  - whether letters were present or not in the ticket didn't matter
  - furthermore, there is too much variance
  - i will not use this feature to reduce noise and focus on more important features

Visualising Name variable

```
In [34]: print("Number of unique names: ",len(titanic_data['Name'].unique()))
```

```
Number of unique names:  891
```

```
In [35]: titanic_data['Name'].unique()
```

```
Out[35]: array(['Braund, Mr. Owen Harris',
       'Cumings, Mrs. John Bradley (Florence Briggs Thayer)',
       'Heikkinen, Miss. Laina',
       'Futrelle, Mrs. Jacques Heath (Lily May Peel)',
       'Allen, Mr. William Henry', 'Moran, Mr. James',
       'McCarthy, Mr. Timothy J', 'Palsson, Master. Gosta Leonard',
       'Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)',
       'Nasser, Mrs. Nicholas (Adele Achem)',
       'Sandstrom, Miss. Marguerite Rut', 'Bonnell, Miss. Elizabeth',
       'Saundercock, Mr. William Henry', 'Andersson, Mr. Anders Johan',
       'Vestrom, Miss. Hulda Amanda Adolfina',
       'Hewlett, Mrs. (Mary D Kingcome) ', 'Rice, Master. Eugene',
       'Williams, Mr. Charles Eugene',
       'Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)',
       'Masselmani, Mrs. Fatima', 'Fynney, Mr. Joseph J',
       'Beesley, Mr. Lawrence', 'McGowan, Miss. Anna "Annie"',
       'Sloper, Mr. William Thompson', 'Palsson, Miss. Torborg Danira',
       'Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)',
       'Emir, Mr. Farred Chehab', 'Fortune, Mr. Charles Alexander',
```

- insight:
  - looking at the Name column, there seems to be a some prefixes:

        - Master
        - Mr
        - Miss (and etc.)
    - they always end with a fullstop and begin with a space

In [36]:
```
def extract_prefixes(target):
    temp=target.split('.')[0]
    return temp.split(' ')[-1]
```

In [37]:
```
titanic_data['Name'].apply(lambda x: extract_prefixes(x)).value_counts()
```

Out[37]:
```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Major         2
Col           2
Mlle          2
Jonkheer      1
Lady          1
Capt          1
Sir           1
Don           1
Ms            1
Mme           1
Countess      1
Name: Name, dtype: int64
```

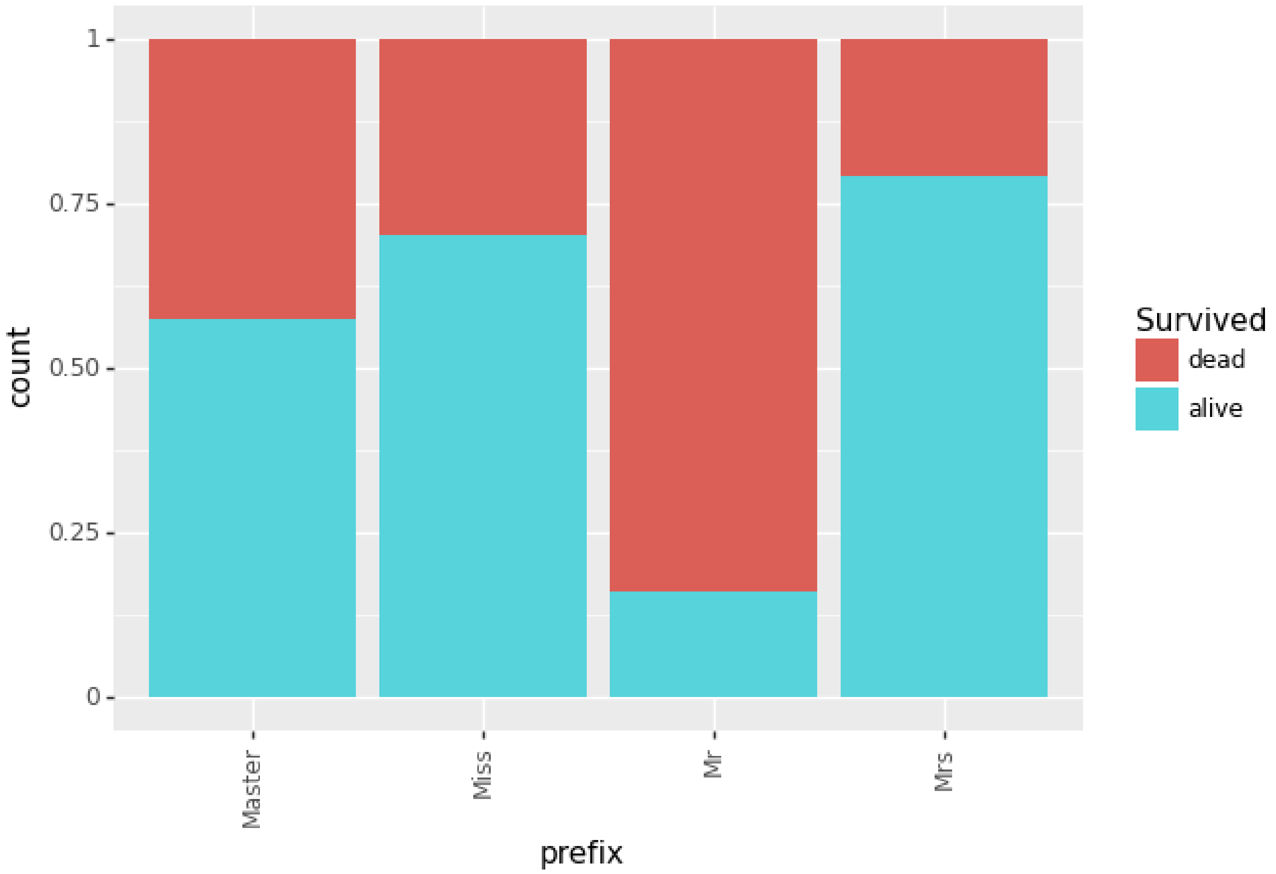
- Decision:
    - i will group rare levels together based on the gender implied by their name prefixes
    - also, i will keep Miss and Mrs as the same level

In [38]:
```
titanic_data['prefix']=titanic_data['Name'].apply(lambda x: extract_prefixes(x))
```

In [39]:
```
def reclassify_prefix_by_gender(target):
    if target in ['Mlle', 'Ms', 'Countess', 'Jonkheer', 'Mme','Lady']:
        return 'Miss'
    elif target in ['Miss','Mr','Master','Mrs']:
        return target
    else:
        return 'Mr'
```

In [40]:
```
titanic_data['prefix']=titanic_data['prefix'].apply(lambda x: reclassify_prefix_by_gend
```

In [41]:
```
ggplot(titanic_data,aes(x='prefix',fill='Survived'))+geom_bar(position='fill')+  theme(
```

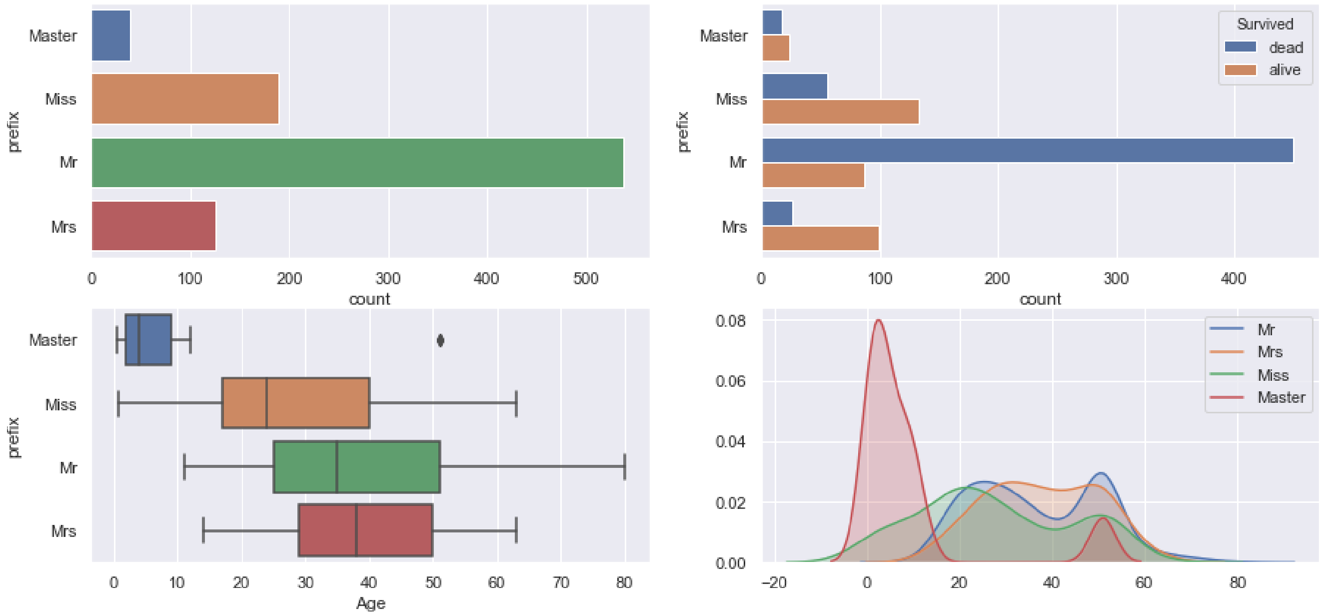


Out[41]: <ggplot: (-9223371963726265808)>

- Observations:
  - i suspect this feature will yield the same effect as gender but slightly better because it has the added level 'Master' which is basically younger Males

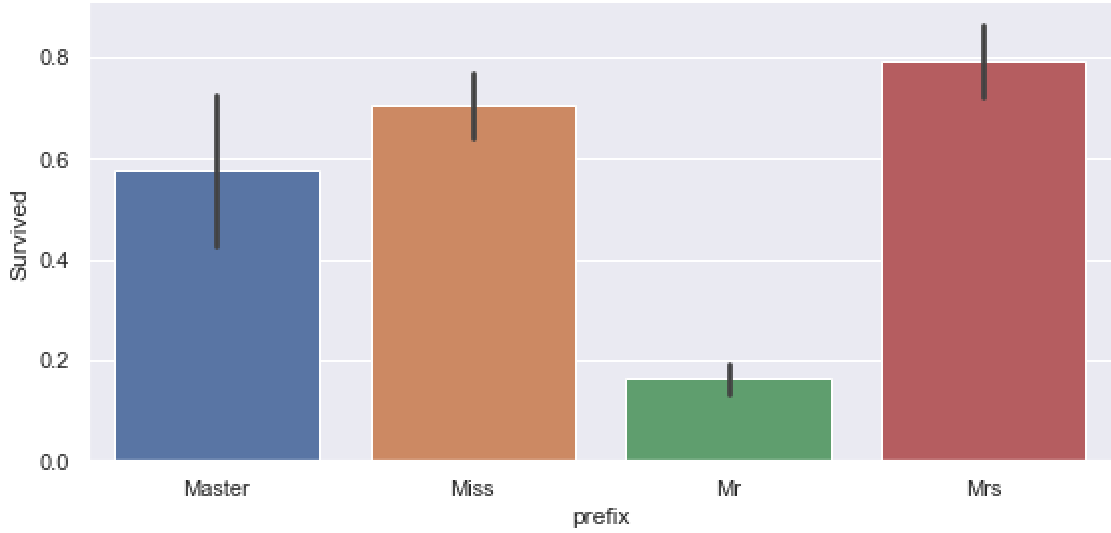
In [42]:
```
titanic_data['prefix']=pd.Categorical(titanic_data['prefix'])
```

In [43]:
```
f, axes = plt.subplots(2,2 , figsize=(15, 7))
sb.countplot(data=titanic_data,y='prefix',ax=axes[0,0])
sb.countplot(data=titanic_data,y='prefix',hue='Survived',ax=axes[0,1])
sb.boxplot(data=titanic_data,x='Age',y='prefix',ax=axes[1,0])
for level in titanic_data['prefix'].unique():
    sb.kdeplot(titanic_data[titanic_data['prefix']==level]['Age'],ax=axes[1,1],shade=Tr
```



In [44]:
```
temp_data=pd.concat([titanic_data['prefix'],response_copy],axis=1)
sb.catplot(x='prefix',y='Survived',data=temp_data,kind='bar',height=4, aspect=2)
```

Out[44]: <seaborn.axisgrid.FacetGrid at 0x110694fc588>



- Observation:
  - prefix is not only a good predictor for survival('Mr' are more likely to die)
  - it also **predicts well for age** as well

Data Preparation

In [233]:
```
# Basic Libraries
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt # we only need pyplot
from plotnine import *
sb.set() # set the default Seaborn style for graphics
from imblearn.over_sampling import ADASYN,BorderlineSMOTE
```

In [234]:
```
# read data
titanic_train=pd.read_csv('train.csv')
titanic_test=pd.read_csv('test.csv')
print('Training set dimensions:',titanic_train.shape)
print('Test set dimensions:',titanic_test.shape)
```

```
Training set dimensions: (891, 12)
Test set dimensions: (418, 11)
```

In [235]:
```
# combine training and test set to do processing together
titanic_processing=pd.concat([titanic_train,titanic_test],axis=0,sort=True).reset_index
```

In [236]:
```
titanic_processing.isna().sum()
```

Out[236]:
```
Age             263
Cabin          1014
Embarked          2
Fare              1
Name              0
Parch             0
PassengerId       0
Pclass            0
Sex               0
SibSp             0
Survived        418
Ticket            0
dtype: int64
```

- Observation:
  - NAs present in 4 features
  - NAs in Survived refer to the test set

In [237]:
```
titanic_processing.dtypes
```

Out[237]:
```
Age            float64
Cabin           object
Embarked        object
Fare           float64
Name            object
Parch            int64
PassengerId      int64
Pclass           int64
Sex             object
SibSp            int64
Survived       float64
Ticket          object
dtype: object
```

variable: Pclass

In [238]:
```
def WoE_encoder(temp_data,columns,response):


    for col in columns:
        # create the WoE encoder
        woe_df = temp_data.groupby([col]).mean()
        woe_df = woe_df.rename(columns={response:'Event'})
        woe_df["Non_Event"]=1-woe_df.Event
        # avoid division by 0
        woe_df["Non_Event"]=np.where(woe_df["Non_Event"]==0,0.00001,woe_df["Non_Event"]
        woe_df["Event"]=np.where(woe_df["Event"]==0,0.00001,woe_df["Event"])
        # np.log is actually ln
        woe_df["WoE"]=np.log(woe_df.Event/woe_df.Non_Event)

        # map to the desired variable
        temp_data[col+'_WoE_encoded']=temp_data[col].map(woe_df.WoE)

    return temp_data
```

In [239]:
```
titanic_processing=WoE_encoder(temp_data=titanic_processing,columns=['Pclass'],response
```

- Steps:
  - **Weight of Evidence encoding** applied
    - works well on binary classification

variable: Name , engineered feature: prefix

In [240]:
```
def extract_prefixes(target):
    temp=target.split('.')[0]
    return temp.split(' ')[-1]
def reclassify_prefix_by_gender(target):
    if target in ['Mlle', 'Ms', 'Countess', 'Jonkheer', 'Mme','Lady']:
        return 'Mrs'
    elif target in ['Mr','Master', 'Mrs','Miss']:
        return target
    else:
        return 'Mr'
titanic_processing['prefix']=titanic_processing['Name'].apply(lambda x: extract_prefixe
titanic_processing['prefix']=titanic_processing['prefix'].apply(lambda x: reclassify_pr
titanic_processing.drop(['Name'],axis=1,inplace=True)
```

- Steps:
  - engineered categorical feature; prefix, from name variable:
    - classes: Mr, Mrs, Master, Miss
    - the remaining prefixes are quite rare so i will group they to Mr and Mrs respectively(by gender)
  - drop original feature

In [241]:
```
titanic_processing=WoE_encoder(temp_data=titanic_processing,columns=['prefix'],response
```

variable: Sex

In [242]:
```
titanic_processing=WoE_encoder(temp_data=titanic_processing,columns=['Sex'],response='S
```

- Steps:
  - WoE encoded

variable: SibSp and Parch , engineered feature: family_size

In [243]:
```
titanic_processing['family_size']=titanic_processing['SibSp']+titanic_processing['Parch
```

- steps:
  - created a variable that is a combination of sibsp and parch

variable: Ticket

In [244]:
```
titanic_processing.drop(['Ticket'],axis=1,inplace=True)
```

- Steps:
  - ticket dropped because it is not predictive of survival

variable: Cabin

In [245]:
```
# fill NAs with a new category 'Missing'
titanic_processing['Cabin'].fillna('Missing',inplace=True)
```

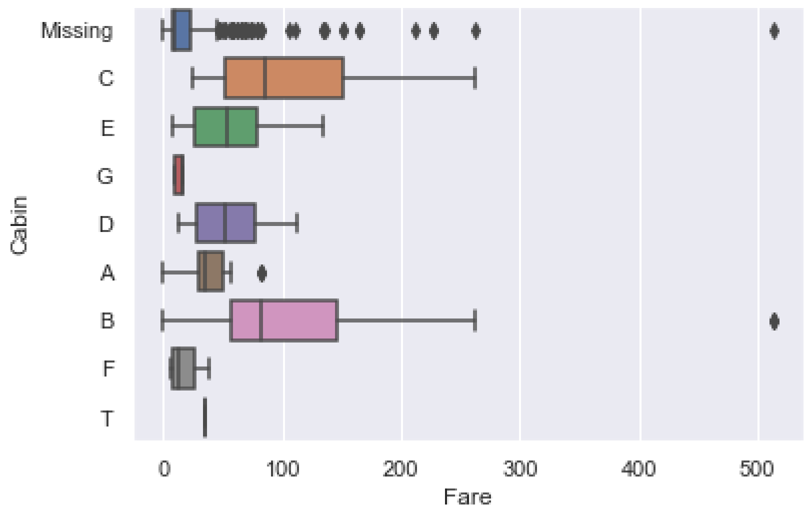
In [246]:
```
# grouped cabins by their alphabets
titanic_processing['Cabin']=titanic_processing['Cabin'].apply(lambda x: x if x=='Missin
```

In [247]:
```
# WoE encode
titanic_processing=WoE_encoder(temp_data=titanic_processing,columns=['Cabin'],response=
```

variable: fare

In [248]:
```
sb.boxplot(data=titanic_processing,y='Cabin',x='Fare')
```

Out[248]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1fb6c1c08>



- Observations:
  - Cabin does seem to be a strong indicator of Fare

In [249]:
```
titanic_processing[titanic_processing['Fare'].isna()]
```

Out[249]:

	Age	Cabin	Embarked	Fare	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Pclass_WoE_enc
1043	60.5	Missing	S	NaN	0	1044	3	male	0	NaN	-1.1

In [250]:
```
titanic_processing.loc[titanic_processing['Fare'].isna(),'Fare']=titanic_processing.loc
```

- Steps:
  - Replace Fare NAs with median based on Cabin

variable: Embarked

In [251]:
```
titanic_processing[titanic_processing['Embarked'].isna()]
```

Out[251]:

	Age	Cabin	Embarked	Fare	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Pclass_WoE_enc
61	38.0	B	NaN	80.0	0	62	1	female	0	1.0	0.53
829	62.0	B	NaN	80.0	0	830	1	female	0	1.0	0.53

In [252]:
```
# replace NAs with mode
titanic_processing['Embarked'].fillna(titanic_processing['Embarked'].mode()[0],inplace=
```

In [253]:
```
# WoE encode
titanic_processing=WoE_encoder(temp_data=titanic_processing,columns=['Embarked'],respon
```

variable: Age

In [254]:
```
# filled NAs with by median of based on 'prefix' variable
for level in titanic_processing['prefix'].unique():
    median_age=titanic_processing[titanic_processing['prefix']==level]['Age'].median()
    titanic_processing.loc[titanic_processing['prefix']==level,'Age']=titanic_processin
```

- IMPT note:
  - titanic_processing.loc[titanic_processing['prefix']==level,'Age'] works because it's a single operation -titanic_processing[titanic_processing['prefix']==level]['Age'] will not because it's a chained operation which will give a warning

variable: PassengerID

In [255]:
```
titanic_processing.drop(['PassengerId'],axis=1,inplace=True)
```

- Steps:
  - dropped

In [256]:
```
# woe=WOEEncoder(cols=['Cabin','Embarked','Sex','Pclass','prefix'])
# woe_train_set=titanic_processing[~titanic_processing.Survived.isna()]
# woe.fit(woe_train_set,woe_train_set.Survived)
# titanic_processing=woe.transform(titanic_processing)
```

In [257]:
```
# LOO=LeaveOneOutEncoder(cols=['Cabin','Embarked','Sex','Pclass','prefix'])
# LOO_train_set=titanic_processing[~titanic_processing.Survived.isna()]
# LOO.fit(woe_train_set,woe_train_set.Survived)
# titanic_processing=LOO.transform(titanic_processing)
```

In [258]:
```
ordinal=OrdinalEncoder(cols=['Cabin','Embarked','Sex','Pclass','prefix'])
ordinal_train_set=titanic_processing[~titanic_processing.Survived.isna()]
ordinal.fit(woe_train_set,woe_train_set.Survived)
titanic_processing=ordinal.transform(titanic_processing)
```

consolidating cleaned data

In [259]:
```
relevant_features=['Age','Cabin','Embarked','Fare', 'Parch',
                   'Pclass','Sex', 'family_size', 'SibSp','prefix','Survived']
```

In [260]:
```
titanic_processing[relevant_features].dtypes
```

Out[260]:
```
Age            float64
Cabin            int32
Embarked         int32
Fare           float64
Parch            int64
Pclass           int32
Sex              int32
family_size      int64
SibSp            int64
prefix           int32
Survived       float64
dtype: object
```

In [261]:
```
# scale the numeric variables to speed up modelling
from sklearn.preprocessing import MinMaxScaler
MMscaler = MinMaxScaler()
titanic_processing[['Age', 'Fare']]=MMscaler.fit_transform(titanic_processing[['Age', '
```

In [262]:
```
titanic_cleaned=titanic_processing[relevant_features].copy()
```

In [263]:
```
# check data types
titanic_cleaned.dtypes
```

Out[263]:
```
Age            float64
Cabin            int32
Embarked         int32
Fare           float64
Parch            int64
Pclass           int32
Sex              int32
family_size      int64
SibSp            int64
prefix           int32
Survived       float64
dtype: object
```

In [264]:
```
# split back to training and test sets
titanic_test_cleaned=titanic_cleaned[titanic_cleaned['Survived'].isna()]
titanic_train_cleaned=titanic_cleaned[~titanic_cleaned['Survived'].isna()]
```

In [265]:
```
# convert response variable to categorical
titanic_train_cleaned.loc[:,'Survived']=pd.Categorical(titanic_train_cleaned['Survived'
```

```
C:\Users\tanch\Anaconda3\lib\site-packages\pandas\core\indexing.py:494: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self.obj[item] = s
```

In [266]:
```
y_titanic_train_cleaned=titanic_train_cleaned['Survived']
X_titanic_train_cleaned=titanic_train_cleaned.drop(['Survived'],axis=1)
X_titanic_test_cleaned=titanic_test_cleaned.drop(['Survived'],axis=1)
```

- Steps:
  - separate predictor from response variable

apply Borderline Smote

- since there is a slight class imbalance i will apply smote so the model can classify minority classes better
- an improved version of SMOTE that only over samples minority points near the decision boundary

In [267]:
```
# oversample = BorderlineSMOTE()
# X, y = oversample.fit_resample(X_titanic_train_cleaned, y_titanic_train_cleaned)
```

In [268]:
```
X, y = X_titanic_train_cleaned,y_titanic_train_cleaned
```

Training the models

In [269]:
```
from catboost import CatBoostClassifier
```

In [42]:
```
model = CatBoostClassifier(loss_function='Logloss')
grid = {'learning_rate': [0.03, 0.1, 0.5],
        'depth': [4, 6, 8],
        'l2_leaf_reg': [1,5,9]}
```

In [43]:
```python
grid_search_result = model.grid_search(grid,
                                       X=X,
                                       y=y)
```

```
0:      loss: 0.4005855 best: 0.4005855 (0)     total: 3s       remaining: 1m 18s
1:      loss: 0.3950105 best: 0.3950105 (1)     total: 6.04s    remaining: 1m 15s
2:      loss: 0.4246609 best: 0.3950105 (1)     total: 8.59s    remaining: 1m 8s
3:      loss: 0.4048127 best: 0.3950105 (1)     total: 11.4s    remaining: 1m 5s
4:      loss: 0.4048486 best: 0.3950105 (1)     total: 14.1s    remaining: 1m 2s
5:      loss: 0.4039558 best: 0.3950105 (1)     total: 16.9s    remaining: 59.1s
6:      loss: 0.4098942 best: 0.3950105 (1)     total: 19.8s    remaining: 56.5s
7:      loss: 0.4112215 best: 0.3950105 (1)     total: 22.6s    remaining: 53.6s
8:      loss: 0.4040524 best: 0.3950105 (1)     total: 25.1s    remaining: 50.2s
9:      loss: 0.4140927 best: 0.3950105 (1)     total: 28.9s    remaining: 49.1s
10:     loss: 0.4252355 best: 0.3950105 (1)     total: 32.6s    remaining: 47.5s
11:     loss: 0.4514362 best: 0.3950105 (1)     total: 36.3s    remaining: 45.3s
12:     loss: 0.4150437 best: 0.3950105 (1)     total: 39.9s    remaining: 43s
13:     loss: 0.4246404 best: 0.3950105 (1)     total: 43.6s    remaining: 40.5s
14:     loss: 0.4145524 best: 0.3950105 (1)     total: 47.3s    remaining: 37.8s
15:     loss: 0.4107085 best: 0.3950105 (1)     total: 50.8s    remaining: 34.9s
16:     loss: 0.4082532 best: 0.3950105 (1)     total: 54.5s    remaining: 32.1s
17:     loss: 0.4058823 best: 0.3950105 (1)     total: 58.1s    remaining: 29.1s
18:     loss: 0.4285091 best: 0.3950105 (1)     total: 1m 5s    remaining: 27.4s
```

In [45]:
```python
# best parameters
grid_search_result['params']
```

Out[45]:
```
{'depth': 4, 'l2_leaf_reg': 1, 'learning_rate': 0.1}
```

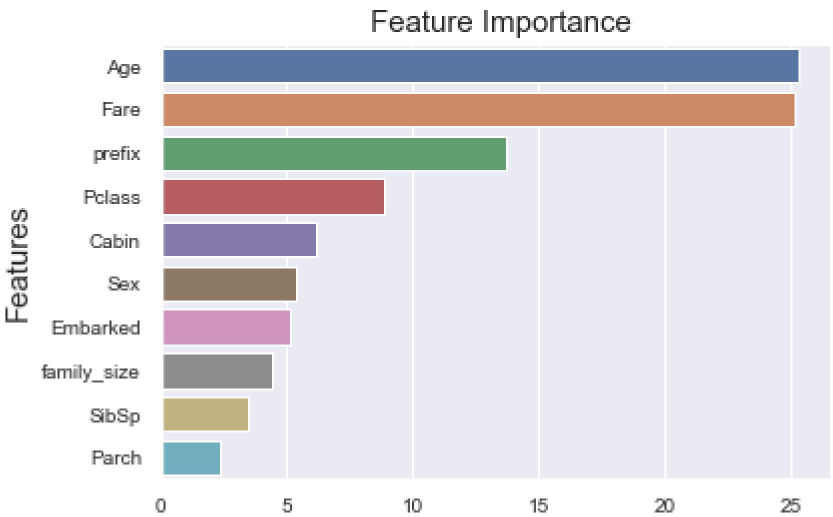
In [270]:
```python
# train catboost on the best parameters
X=pd.DataFrame(X,columns=X_titanic_train_cleaned.columns)
catboost = CatBoostClassifier(loss_function='Logloss',
                              depth=8,
                              l2_leaf_reg=1,
                              learning_rate=0.1,
                              iterations=2000)
catboost.fit(X,y)
```

```
0:      learn: 0.6254313        total: 5.54ms   remaining: 11.1s
1:      learn: 0.5799697        total: 7.81ms   remaining: 7.81s
2:      learn: 0.5315329        total: 11.4ms   remaining: 7.61s
3:      learn: 0.4965131        total: 13.3ms   remaining: 6.66s
4:      learn: 0.4658958        total: 17.2ms   remaining: 6.88s
5:      learn: 0.4460667        total: 19.2ms   remaining: 6.37s
6:      learn: 0.4263617        total: 22.7ms   remaining: 6.47s
7:      learn: 0.4186640        total: 23.9ms   remaining: 5.95s
8:      learn: 0.4087710        total: 27.6ms   remaining: 6.12s
9:      learn: 0.3938328        total: 31.2ms   remaining: 6.2s
10:     learn: 0.3911457        total: 32.4ms   remaining: 5.85s
11:     learn: 0.3786118        total: 36ms     remaining: 5.96s
12:     learn: 0.3695648        total: 39.7ms   remaining: 6.06s
13:     learn: 0.3624794        total: 43.1ms   remaining: 6.12s
14:     learn: 0.3532192        total: 46.4ms   remaining: 6.14s
15:     learn: 0.3437947        total: 50ms     remaining: 6.2s
16:     learn: 0.3391337        total: 53.5ms   remaining: 6.24s
17:     learn: 0.3336272        total: 57.2ms   remaining: 6.3s
18:     learn: 0.3304653        total: 60.7ms   remaining: 6.33s
```

Feature importance

In [271]:
```
indices=np.argsort(catboost.feature_importances_)[::-1]
g=sb.barplot(y=X.columns[indices],x=catboost.feature_importances_[indices])
g.set_ylabel('Features',fontsize=15)
g.tick_params(labelsize=10)
g.set_title(' Feature Importance',fontsize=15)
```

Out[271]: `Text(0.5, 1.0, ' Feature Importance')`



In [230]:
```

```

Out[230]:

	Age	Cabin	Embarked	Fare	Parch	Pclass	Sex	family_size	SibSp	prefix
0	0.273456	1.0	1.00000	0.014151	0.0	1.000000	1.0	2.0	1.0	1.0
1	0.473882	2.0	2.00000	0.139136	0.0	2.000000	2.0	2.0	1.0	2.0
2	0.323563	1.0	1.00000	0.015469	0.0	1.000000	2.0	1.0	0.0	3.0
3	0.436302	2.0	1.00000	0.103644	0.0	2.000000	2.0	2.0	1.0	2.0
4	0.436302	1.0	1.00000	0.015713	0.0	1.000000	1.0	1.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...
1093	0.244525	1.0	2.23452	0.028691	0.0	1.000000	2.0	2.0	1.0	2.0
1094	0.373669	1.0	1.00000	0.016037	0.0	1.000000	1.0	1.0	0.0	1.0
1095	0.336089	1.0	1.00000	0.013782	0.0	1.000000	1.0	1.0	0.0	1.0
1096	0.398722	1.0	1.00000	0.110272	0.0	1.000000	1.0	1.0	0.0	1.0
1097	0.442603	1.0	1.00000	0.130680	1.0	1.321641	2.0	3.0	1.0	2.0

1098 rows × 10 columns

In [147]:
```
# make predictions
y_pred=pd.Series(catboost.predict(X_titanic_test_cleaned),name='Survived')
catboost_predictions=pd.concat([titanic_test.PassengerId,y_pred],axis=1)
```

In [148]:
```
# export csv
catboost_predictions.to_csv('catboost_predictions.csv',index=False)
```

RESULTS

In [123]:
```python
from IPython.display import Image
```

In [124]:
```python
Image(filename='catboost_scores.JPG')
```

Out[124]: