ICEI-Tutorial

February 2, 2017

```
In [1]: import numpy as np
        import pandas as pd
        from patsy import dmatrices
        import statsmodels.api as sm
        from sklearn import svm
        import warnings
        import matplotlib.pyplot as plt

        # To display plots inside notebook
        %matplotlib inline

        warnings.filterwarnings('ignore')

        # notebook parameters
        pd.set_option('display.max_rows', 15)
```

0.0.1 Data Handling

Let's read our data in using pandas:

```
In [2]: df = pd.read_csv(r"data/train.csv")

In [3]: df

Out[3]:      PassengerId  Survived  Pclass  \
        0              1         0       3
        1              2         1       1
        2              3         1       3
        3              4         1       1
        4              5         0       3
        5              6         0       3
        6              7         0       1
        ..           ...       ...     ...
        884          885         0       3
        885          886         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  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
5                                     Moran, Mr. James    male   NaN      0
6                              McCarthy, Mr. Timothy J    male  54.0      0
..                                                 ...     ...   ...    ...
884                              Sutehall, Mr. Henry Jr    male  25.0      0
885            Rice, Mrs. William (Margaret Norton)  female  39.0      0
886                               Montvila, Rev. Juozas    male  27.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                               Behr, Mr. Karl Howell    male  26.0      0
890                                 Dooley, Mr. Patrick    male  32.0      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
5        0            330877   8.4583   NaN        Q
6        0             17463  51.8625   E46        S
..     ...               ...      ...   ...      ...
884      0   SOTON/OQ 392076   7.0500   NaN        S
885      5            382652  29.1250   NaN        Q
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

To view the columns individually

In [4]:
```
df['Name']
```

Out[4]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
```

```
5                                              Moran, Mr. James
6                                      McCarthy, Mr. Timothy J
                                   ...
884                                     Sutehall, Mr. Henry Jr
885                       Rice, Mrs. William (Margaret Norton)
886                                      Montvila, Rev. Juozas
887                               Graham, Miss. Margaret Edith
888                   Johnston, Miss. Catherine Helen "Carrie"
889                                      Behr, Mr. Karl Howell
890                                        Dooley, Mr. Patrick
Name: Name, dtype: object
```

To find the occurence of each object

In [5]:
```
df['Pclass'].value_counts()
```

Out[5]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [6]:
```
df['Sex'].value_counts()
```

Out[6]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [7]:
```
help(pd.DataFrame.apply)
```

```
Help on method apply in module pandas.core.frame:

apply(self, func, axis=0, broadcast=False, raw=False, reduce=None, args=(), **kwds) unbound pand
    Applies function along input axis of DataFrame.

    Objects passed to functions are Series objects having index
    either the DataFrame's index (axis=0) or the columns (axis=1).
    Return type depends on whether passed function aggregates, or the
    reduce argument if the DataFrame is empty.

    Parameters
    ----------
    func : function
        Function to apply to each column/row
    axis : {0 or 'index', 1 or 'columns'}, default 0
        * 0 or 'index': apply function to each column
        * 1 or 'columns': apply function to each row
    broadcast : boolean, default False
        For aggregation functions, return object of same size with values
        propagated
```

```
    raw : boolean, default False
        If False, convert each row or column into a Series. If raw=True the
        passed function will receive ndarray objects instead. If you are
        just applying a NumPy reduction function this will achieve much
        better performance
    reduce : boolean or None, default None
        Try to apply reduction procedures. If the DataFrame is empty,
        apply will use reduce to determine whether the result should be a
        Series or a DataFrame. If reduce is None (the default), apply's
        return value will be guessed by calling func an empty Series (note:
        while guessing, exceptions raised by func will be ignored). If
        reduce is True a Series will always be returned, and if False a
        DataFrame will always be returned.
    args : tuple
        Positional arguments to pass to function in addition to the
        array/series
    Additional keyword arguments will be passed as keywords to the function

    Notes
    -----
    In the current implementation apply calls func twice on the
    first column/row to decide whether it can take a fast or slow
    code path. This can lead to unexpected behavior if func has
    side-effects, as they will take effect twice for the first
    column/row.

    Examples
    --------
    >>> df.apply(numpy.sqrt) # returns DataFrame
    >>> df.apply(numpy.sum, axis=0) # equiv to df.sum(0)
    >>> df.apply(numpy.sum, axis=1) # equiv to df.sum(1)

    See also
    --------
    DataFrame.applymap: For elementwise operations

    Returns
    -------
    applied : Series or DataFrame
```

In [8]:
```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[8]:
```
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 [9]:
```
df.apply(lambda x: sum(x.notnull()),axis=0)
```

Out[9]:
```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
dtype: int64
```

To drop the column that is not required

In [10]:
```
df.drop(['Cabin'], axis=1)
```

Out[10]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
5              6         0       3
6              7         0       1
..           ...       ...     ...
884          885         0       3
885          886         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  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
5                                     Moran, Mr. James    male   NaN      0
6                              McCarthy, Mr. Timothy J    male  54.0      0
..                                                 ...     ...   ...    ...
884                              Sutehall, Mr. Henry Jr    male  25.0      0
885               Rice, Mrs. William (Margaret Norton)  female  39.0      0
886                               Montvila, Rev. Juozas    male  27.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                               Behr, Mr. Karl Howell    male  26.0      0
890                                 Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
5        0            330877   8.4583        Q
6        0             17463  51.8625        S
..     ...               ...      ...      ...
884      0   SOTON/OQ 392076   7.0500        S
885      5            382652  29.1250        Q
886      0            211536  13.0000        S
887      0            112053  30.0000        S
888      2        W./C. 6607  23.4500        S
889      0            111369  30.0000        C
890      0            370376   7.7500        Q

[891 rows x 11 columns]
```

In [11]:
```python
df = df.drop(['Cabin'], axis=1)
# df.drop(['Cabin'], axis=1, inplace=True)
```

In [12]:
```python
# Remove NaN values
df = df.dropna()
# df.dropna(inplace=True)
```

In [13]:
```python
df.head()
```

Out[13]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
```

```
4            5          0       3

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

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

Now let's check for the 'notnull' values

In [14]:
```
df.apply(lambda x: sum(x.notnull()),axis=0)
```

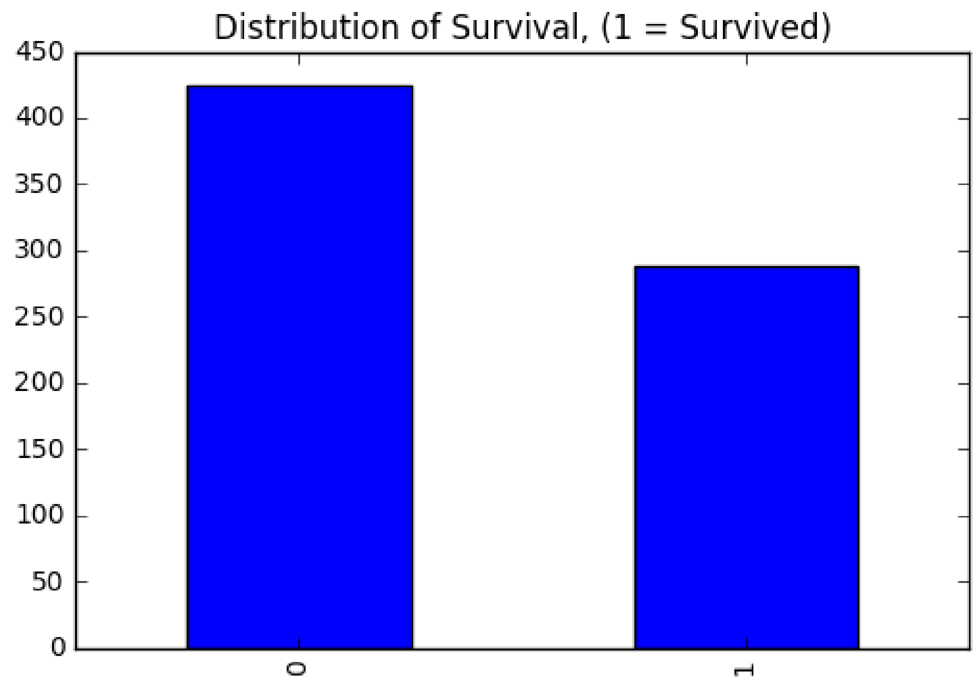
Out[14]:
```
PassengerId    712
Survived       712
Pclass         712
Name           712
Sex            712
Age            712
SibSp          712
Parch          712
Ticket         712
Fare           712
Embarked       712
dtype: int64
```

0.0.2 Visualize our data graphically:

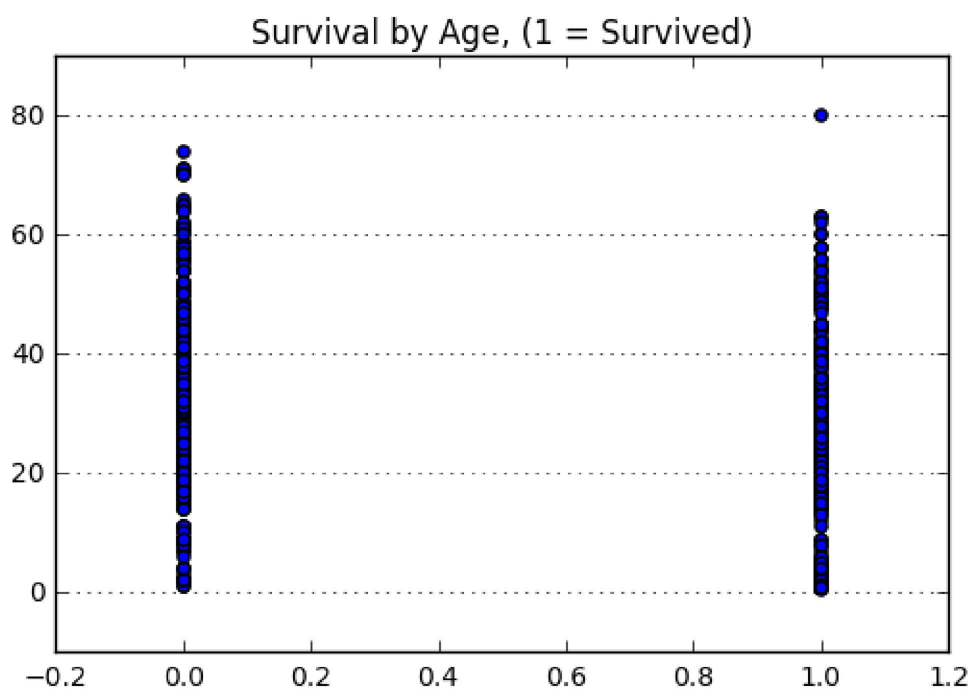
plot a bar graph of those who surived vs those who did not

In [15]:
```
df['Survived'].value_counts().plot(kind='bar')
plt.title("Distribution of Survival, (1 = Survived)")
```

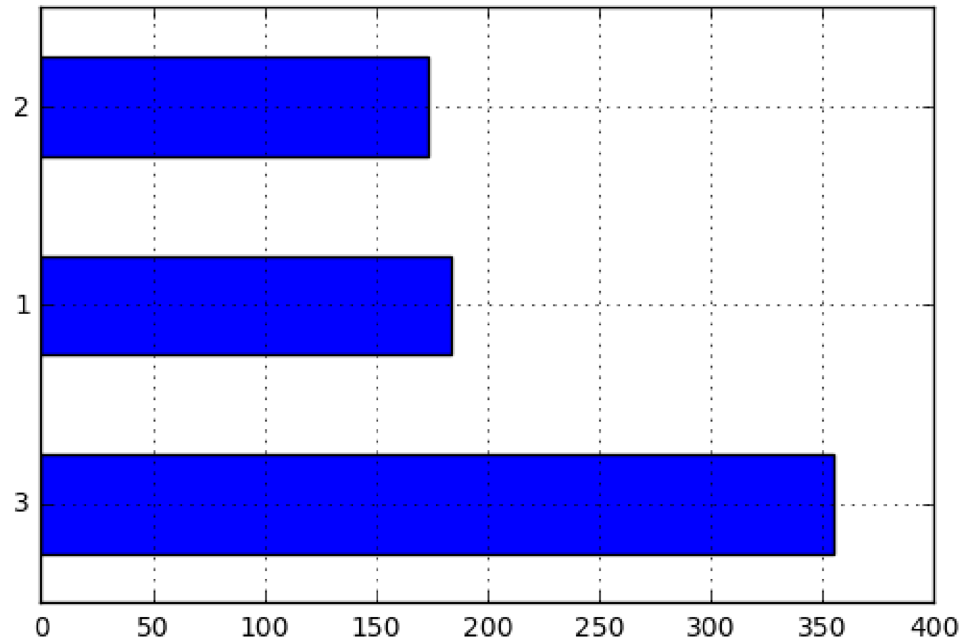
Out[15]:
```
<matplotlib.text.Text at 0x20273b00>
```




```
In [16]: plt.scatter(df.Survived, df.Age)
         plt.title("Survival by Age, (1 = Survived)")
         plt.grid(True, axis='y')
```



In [17]:
```
df.Pclass.value_counts().plot(kind="barh")
plt.grid(True)
```



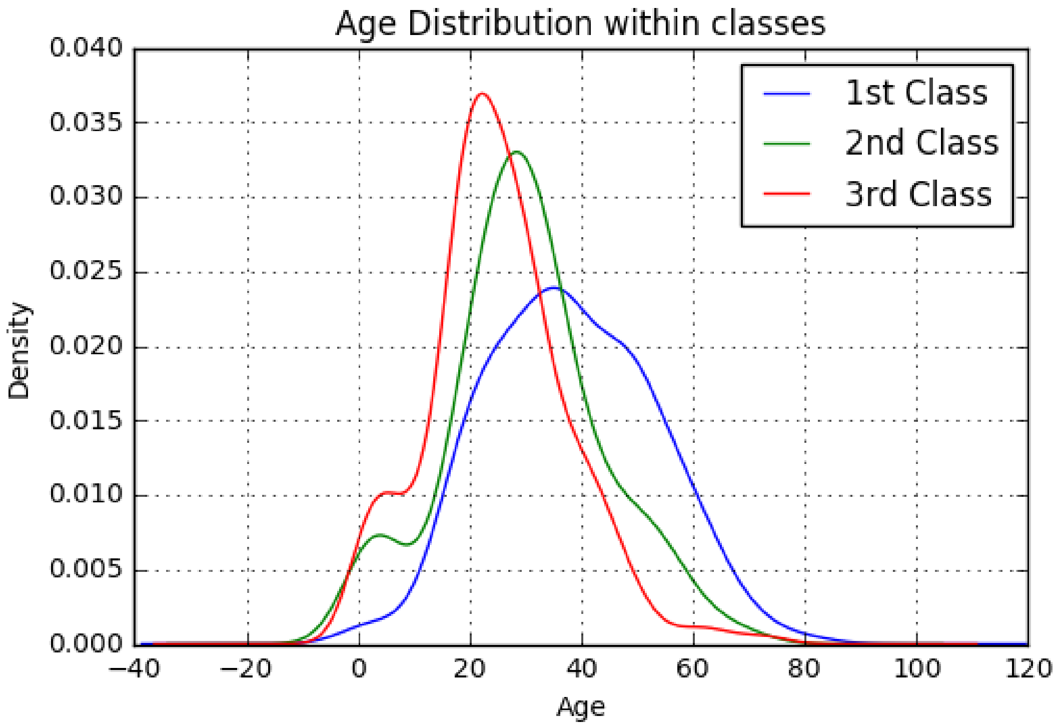
Checking for 'class 1' passengers

In [18]:
```
df['Pclass'] == 1
```

Out[18]:
```
0      False
1       True
2      False
3       True
4      False
6       True
7      False
       ...
883    False
884    False
885    False
886    False
887     True
889     True
890    False
Name: Pclass, dtype: bool
```

Passing the 'class 1' passengers list to 'Age' --> To find out the age of 'class 1' passenges

```
In [19]: df['Age'][df['Pclass'] == 1]

Out[19]: 1      38.0
         3      35.0
         6      54.0
         11     58.0
         23     28.0
         27     19.0
         30     40.0
                ...
         862    48.0
         867    31.0
         871    47.0
         872    33.0
         879    56.0
         887    19.0
         889    26.0
         Name: Age, dtype: float64

In [20]: len(df['Age'][df['Pclass'] == 1])

Out[20]: 184

In [21]: df['Age'][df['Pclass'] == 1].plot(kind='kde')
         df['Age'][df['Pclass'] == 2].plot(kind='kde')
         df['Age'][df['Pclass'] == 3].plot(kind='kde')
         plt.xlabel("Age")
         plt.title("Age Distribution within classes")
         # sets our legend for our graph.
         plt.legend(('1st Class', '2nd Class','3rd Class'),loc='best')
         plt.grid(True)
```

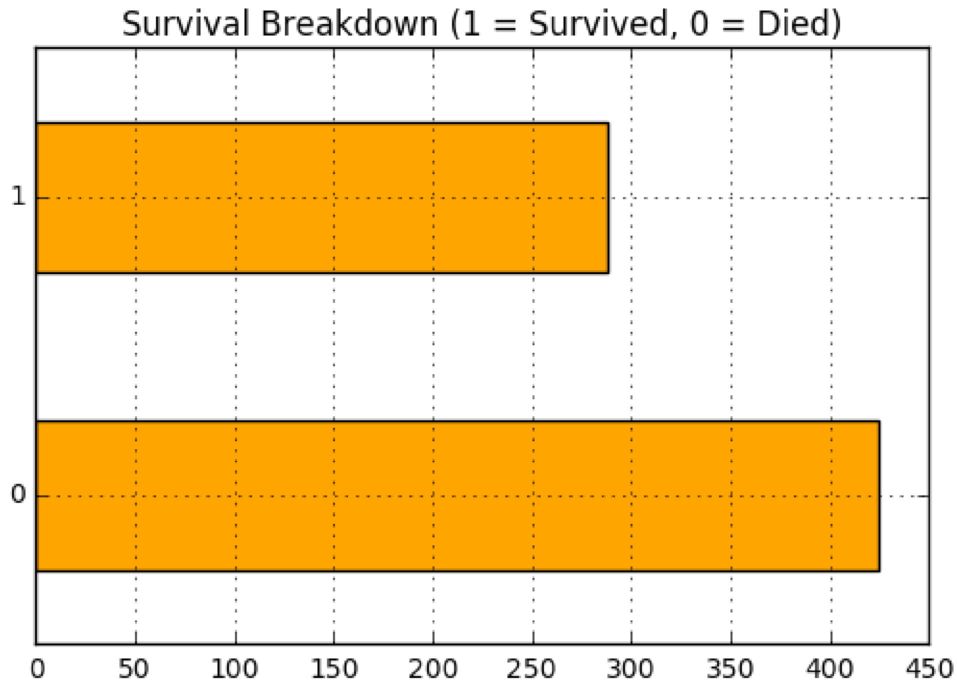


0.0.3 Exploratory Visualization:

The point of this competition is to predict if an individual will survive based on the features in the data like:

- Traveling Class (called pclass in the data)
- Sex
- Age
- Fare Price

```
In [22]: df['Survived'].value_counts().plot(kind='barh', color="orange")
         plt.title("Survival Breakdown (1 = Survived, 0 = Died)")
         plt.grid(True)
```



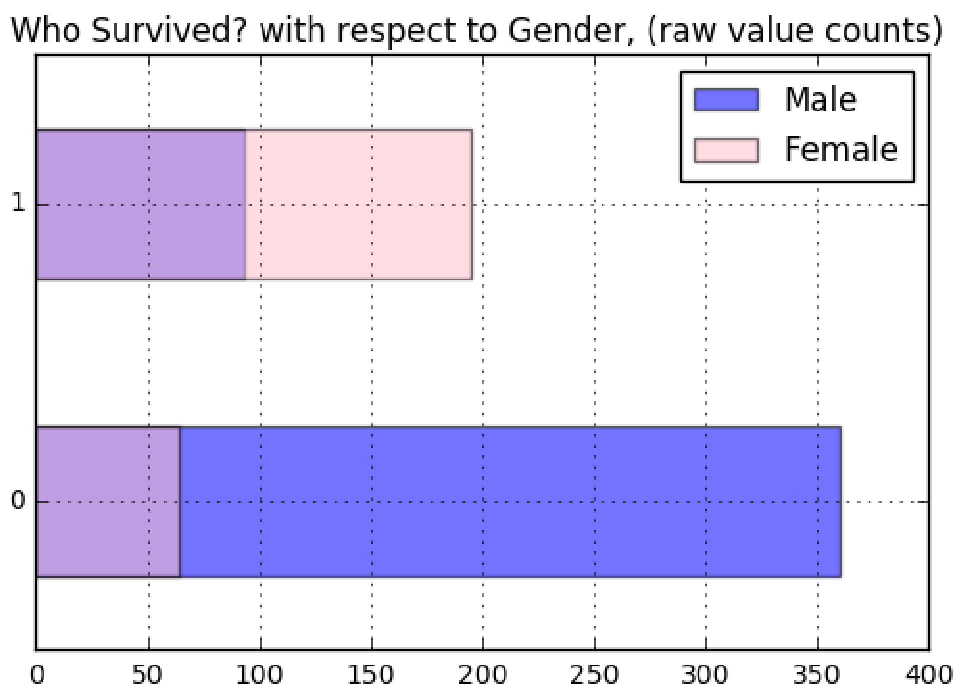
Find out the count of total male and female survived, in ascending order

```
In [23]: df['Survived'][df['Sex'] == 'male'].value_counts()

Out[23]: 0    360
         1     93
         Name: Survived, dtype: int64

In [24]: df['Survived'][df['Sex'] == 'female'].value_counts()

Out[24]: 1    195
         0     64
         Name: Survived, dtype: int64

In [25]: df_male = df['Survived'][df['Sex'] == 'male'].value_counts().sort_index()
         df_female = df['Survived'][df['Sex'] == 'female'].value_counts().sort_index()

In [26]: df_male

Out[26]: 0    360
         1     93
         Name: Survived, dtype: int64

In [27]: df_female
```

```
Out[27]: 0     64
         1    195
         Name: Survived, dtype: int64
```

```
In [28]: df_male.plot(kind='barh', color='blue', label='Male', alpha=0.55)
         df_female.plot(kind='barh', color='pink', label='Female', alpha=0.55)
         plt.grid(True)
         plt.legend(loc='best')
         plt.title("Who Survived? with respect to Gender, (raw value counts) ")
```

```
Out[28]: <matplotlib.text.Text at 0x212243c8>
```



Now let's find out the ratio of survived people

```
In [29]: df_male.sum()
```

```
Out[29]: 453L
```

```
In [30]: df_female.sum()
```

```
Out[30]: 259L
```

```
In [31]: df_male/float(df_male.sum())
```

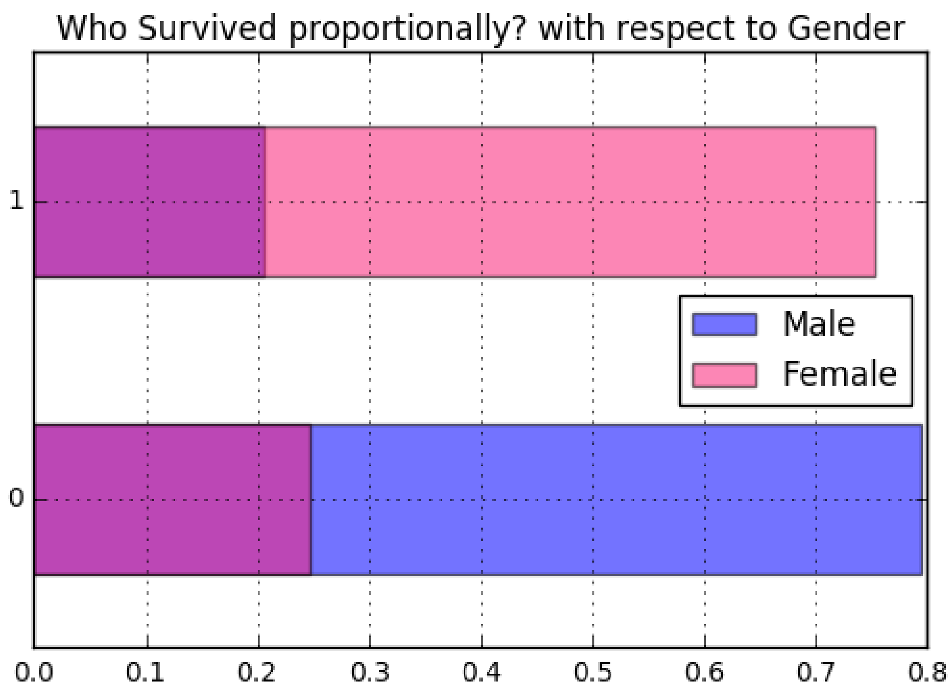
```
Out[31]: 0     0.794702
         1     0.205298
         Name: Survived, dtype: float64
```

```
In [32]: df_female/float(df_female.sum())
```

```
Out[32]: 0    0.247104
         1    0.752896
         Name: Survived, dtype: float64
```

```
In [33]: (df_male/float(df_male.sum())).plot(kind='barh', label='Male', alpha=0.55)
         (df_female/float(df_female.sum())).plot(kind='barh', color='#FA2379', label='Female', a
         plt.title("Who Survived proportionally? with respect to Gender")
         plt.grid(True)
         plt.legend(loc='best')
```

```
Out[33]: <matplotlib.legend.Legend at 0x213dc240>
```



Let's try going some more deeper, by finding out the the passenger class wise survival

```
In [34]: female_highclass = df['Survived'][(df['Pclass'] != 3) & (df['Sex'] == 'female')].value_
         female_lowclass = df['Survived'][(df['Pclass'] == 3) & (df['Sex'] == 'female')].value_c
         male_highclass = df['Survived'][(df['Pclass'] != 3) & (df['Sex'] == 'male')].value_coun
         male_lowclass = df['Survived'][(df['Pclass'] == 3) & (df['Sex'] == 'male')].value_count
```

```
In [35]: female_highclass
```

```
Out[35]: 1    148
         0      9
         Name: Survived, dtype: int64
```

In [36]: female_lowclass

Out[36]:
```
0    55
1    47
Name: Survived, dtype: int64
```

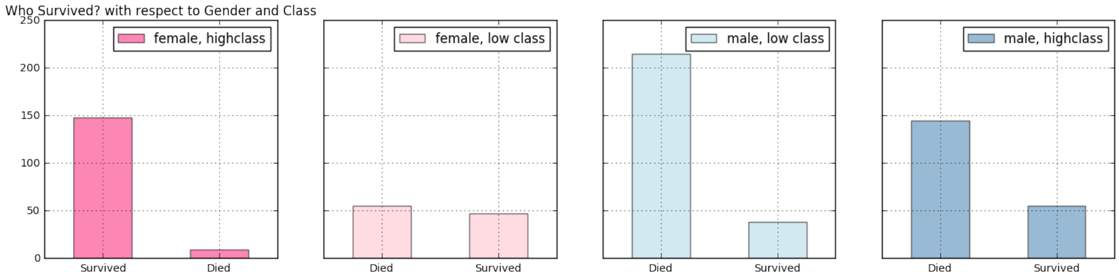
In [37]: male_highclass

Out[37]:
```
0    145
1     55
Name: Survived, dtype: int64
```

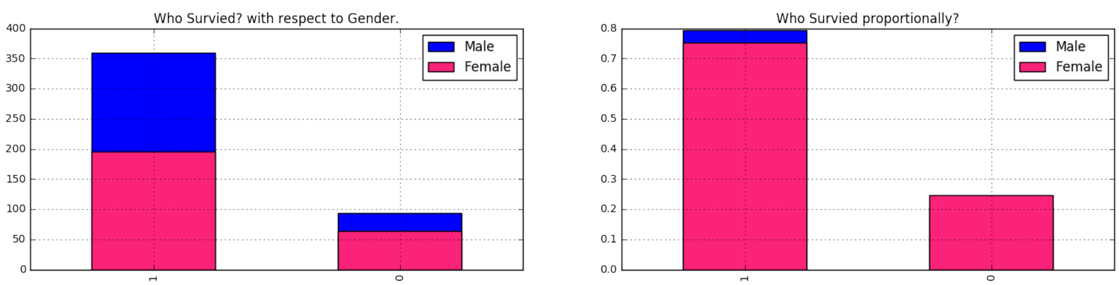
In [38]: male_lowclass

Out[38]:
```
0    215
1     38
Name: Survived, dtype: int64
```

In [39]:
```
fig = plt.figure(figsize=(18,4), dpi=1600)

ax1=fig.add_subplot(141)
female_highclass.plot(kind='bar', label='female, highclass', color='#FA2479', alpha=0.5
ax1.set_xticklabels(["Survived", "Died"], rotation=0)
plt.title("Who Survived? with respect to Gender and Class")
plt.legend(loc='best')
plt.grid(True)

ax2=fig.add_subplot(142, sharey=ax1)
female_lowclass.plot(kind='bar', label='female, low class', color='pink', alpha=0.55)
ax2.set_xticklabels(["Died","Survived"], rotation=0)
plt.legend(loc='best')
plt.grid(True)

ax3=fig.add_subplot(143, sharey=ax1)
male_lowclass.plot(kind='bar', label='male, low class',color='lightblue', alpha=0.55)
ax3.set_xticklabels(["Died","Survived"], rotation=0)
plt.legend(loc='best')
plt.grid(True)

ax4=fig.add_subplot(144, sharey=ax1)
male_highclass.plot(kind='bar', label='male, highclass', alpha=0.55, color='steelblue')
ax4.set_xticklabels(["Died","Survived"], rotation=0)
plt.legend(loc='best')
plt.grid(True)
```




```
In [40]: fig = plt.figure(figsize=(18,4), dpi=1600)
         ax1 = fig.add_subplot(121)
         df.Survived[df.Sex == 'male'].value_counts().plot(kind='bar',label='Male')
         df.Survived[df.Sex == 'female'].value_counts().plot(kind='bar', color='#FA2379',label='
         plt.title("Who Survied? with respect to Gender.")
         plt.legend(loc='best')
         plt.grid(True)

         ax2 = fig.add_subplot(122)
         (df['Survived'][df['Sex'] == 'male'].value_counts()/float(df['Sex'][df['Sex'] == 'male'
         (df['Survived'][df['Sex'] == 'female'].value_counts()/float(df['Sex'][df['Sex'] == 'fem
         plt.title("Who Survied proportionally?")
         plt.legend(loc='best')
         plt.grid(True)
```



Let's just create a formule for our model

```
In [41]: # Ref: http://patsy.readthedocs.org/en/latest/formulas.html
         formula = 'Survived ~ C(Pclass) + C(Sex) + Age + SibSp  + C(Embarked)'
```

'dmatrices' is used to used to create regression friendly dataframe

```
In [42]: y,X = dmatrices(formula, data=df, return_type='dataframe')
         # instantiate our model
         model = sm.Logit(y, X)
```

```
# fit our model to the training data
res = model.fit()

# save the result for outputing predictions later
result = [res, formula]
res.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.444388
         Iterations 6
```

Out[42]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:               Survived   No. Observations:                  712
Model:                          Logit   Df Residuals:                      704
Method:                           MLE   Df Model:                            7
Date:                Thu, 02 Feb 2017   Pseudo R-squ.:                  0.3414
Time:                        00:50:17   Log-Likelihood:                -316.40
converged:                       True   LL-Null:                       -480.45
                                        LLR p-value:                 5.992e-67
=====================================================================================
                        coef    std err          z      P>|z|      [95.0% Conf. Int.]
-------------------------------------------------------------------------------------
Intercept             4.5423      0.474      9.583      0.000         3.613     5.471
C(Pclass)[T.2]       -1.2673      0.299     -4.245      0.000        -1.852    -0.682
C(Pclass)[T.3]       -2.4966      0.296     -8.422      0.000        -3.078    -1.916
C(Sex)[T.male]       -2.6239      0.218    -12.060      0.000        -3.050    -2.197
C(Embarked)[T.Q]     -0.8351      0.597     -1.398      0.162        -2.006     0.335
C(Embarked)[T.S]     -0.4254      0.271     -1.572      0.116        -0.956     0.105
Age                  -0.0436      0.008     -5.264      0.000        -0.060    -0.027
SibSp                -0.3697      0.123     -3.004      0.003        -0.611    -0.129
=====================================================================================
"""
```

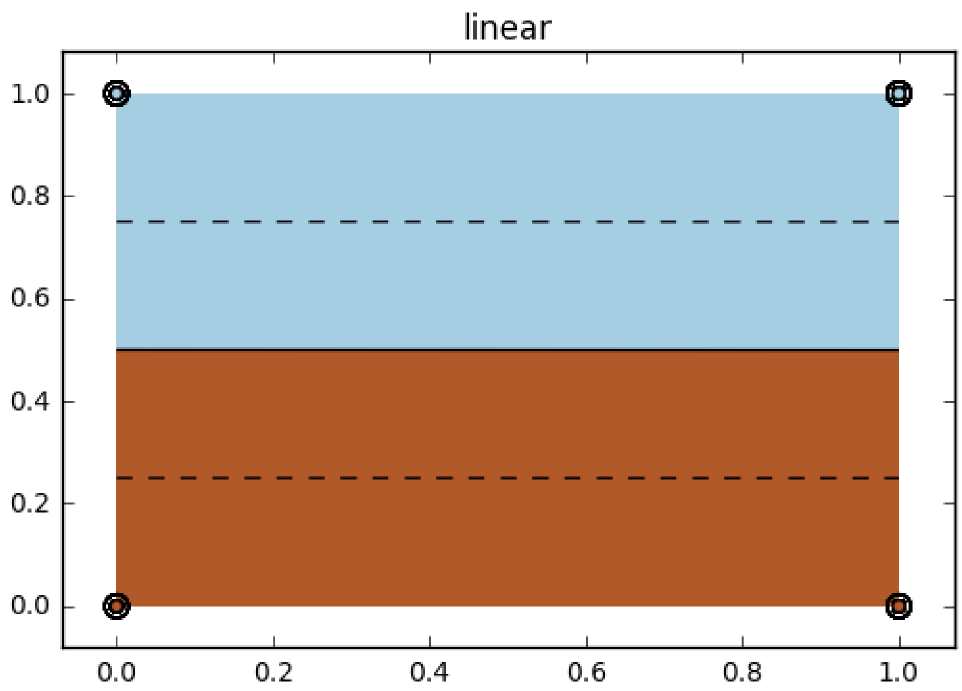
Let's try to do something with machine learning

In [43]:
```
# Create our machine learning formula
formula_ml = 'Survived ~ C(Pclass) + C(Sex) + Age + SibSp + Parch + C(Embarked)'
```

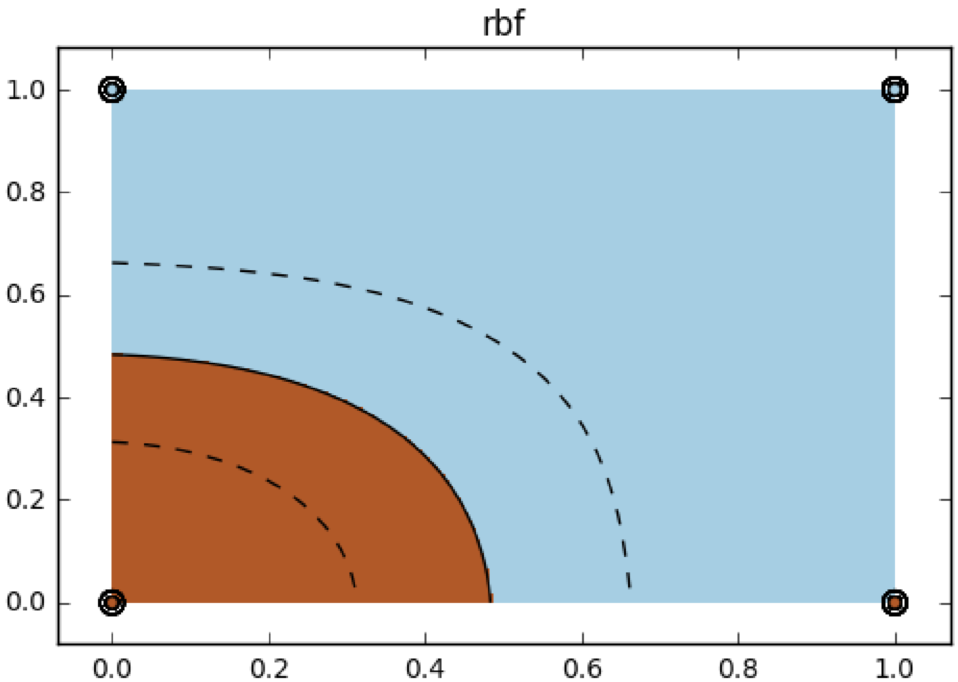
In [44]:
```
# set plotting parameters
plt.figure(figsize=(8,6))

# create a regression friendly data frame
y, x = dmatrices(formula_ml, data=df, return_type='matrix')
```

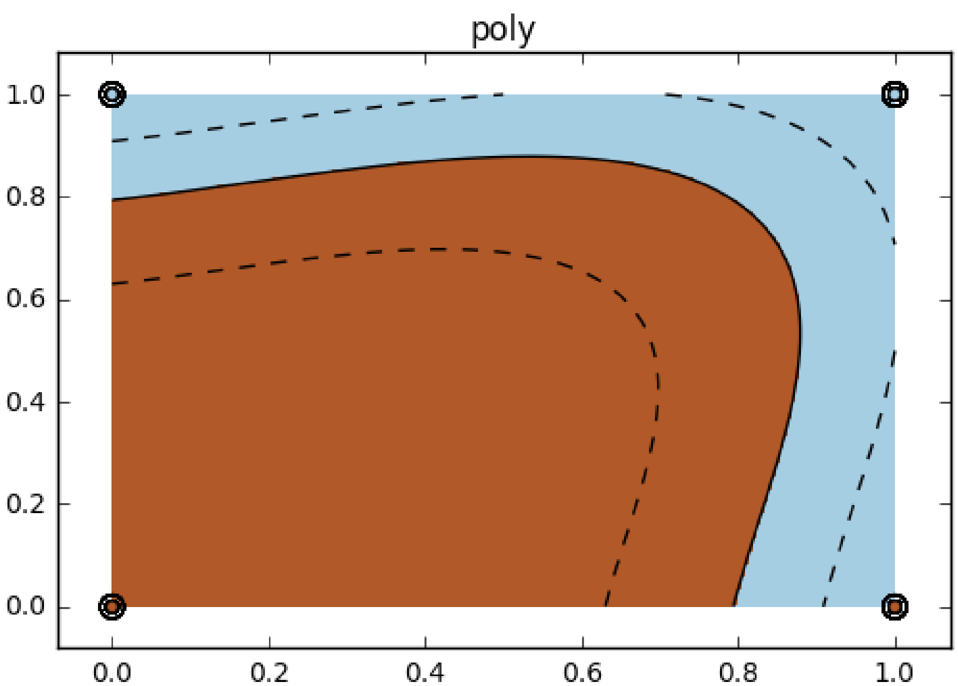
```
# select which features we would like to analyze
# try chaning the selection here for diffrent output.
# Choose : [2,3] - pretty sweet DBs [3,1] --standard DBs [7,3] -very cool DBs,
# [3,6] -- very long complex dbs, could take over an hour to calculate!
feature_1 = 2
feature_2 = 3

X = np.asarray(x)
X = X[:,[feature_1, feature_2]]


y = np.asarray(y)
# needs to be 1 dimenstional so we flatten. it comes out of dmatirces with a shape.
y = y.flatten()

n_sample = len(X)

np.random.seed(0)
order = np.random.permutation(n_sample)

X = X[order]
y = y[order].astype(np.float)

# do a cross validation
nighty_precent_of_sample = int(.9 * n_sample)
X_train = X[:nighty_precent_of_sample]
y_train = y[:nighty_precent_of_sample]
X_test = X[nighty_precent_of_sample:]
y_test = y[nighty_precent_of_sample:]

# create a list of the types of kerneks we will use for your analysis
types_of_kernels = ['linear', 'rbf', 'poly']

# specify our color map for plotting the results
color_map = plt.cm.Paired
# color_map = plt.cm.coolwarm

# fit the model
for fig_num, kernel in enumerate(types_of_kernels):
    clf = svm.SVC(kernel=kernel, gamma=3)
    clf.fit(X_train, y_train)

    plt.figure(fig_num)
    plt.scatter(X[:, 0], X[:, 1], c=y, zorder=10, cmap=color_map)

    # circle out the test data
    plt.scatter(X_test[:, 0], X_test[:, 1], s=80, facecolors='none', zorder=10)
```

```
plt.axis('tight')
x_min = X[:, 0].min()
x_max = X[:, 0].max()
y_min = X[:, 1].min()
y_max = X[:, 1].max()

XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
Z = clf.decision_function(np.c_[XX.ravel(), YY.ravel()])

# put the result into a color plot
Z = Z.reshape(XX.shape)
plt.pcolormesh(XX, YY, Z > 0, cmap=color_map)
plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--', '-', '--'],
            levels=[-.5, 0, .5])

plt.title(kernel)
plt.show()
```




```
<matplotlib.figure.Figure at 0x21308da0>
```






```
In [45]: test_data = pd.read_csv(r"data/test.csv")
```

In [46]:
```
test_data['Survived'] = 1.23
```

In [47]:
```
clf = svm.SVC(kernel='poly', gamma=3).fit(X_train, y_train)
y,x = dmatrices(formula_ml, data=test_data, return_type='dataframe')
```

In [48]:
```
res_svm = clf.predict(x.ix[:,[6,3]].dropna())
```

In [49]:
```
# x.ix[:,[6,3]] considers the 6th and 3rd column, ':' represents all or the entire colu
x.ix[:,[6,3]]
```

Out[49]:
```
      Age  C(Sex)[T.male]
0    34.5             1.0
1    47.0             0.0
2    62.0             1.0
3    27.0             1.0
4    22.0             0.0
5    14.0             1.0
6    30.0             0.0
..    ...             ...
406  23.0             1.0
407  50.0             1.0
409   3.0             0.0
411  37.0             0.0
412  28.0             0.0
414  39.0             0.0
415  38.5             1.0

[332 rows x 2 columns]
```

In [50]:
```
res_svm
```

Out[50]:
```
array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
```

```
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0.])
```

Convert the numpy array to dataframe

In [51]:
```
res_svm = pd.DataFrame(res_svm,columns=['Survived'])
```

In [52]:
```
res_svm
```

Out[52]:
```
     Survived
0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
5         0.0
6         0.0
..        ...
325       0.0
326       0.0
327       0.0
328       0.0
329       0.0
330       0.0
331       0.0

[332 rows x 1 columns]
```

In [53]:
```
clf.predict(np.array([50, 1]))
```

Out[53]:
```
array([ 0.])
```