Titanic

Survived or Not??

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Understanding the data...

```
data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Check the Missing Va

```
data.isnull().sum()
```

```
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Fare             0
Embarked         0
FamilySize       0
IsAlone          0
AgeBin           0
Title            0
FareBin          0
Sex_Code         0
Title_Code       0
AgeBin_Code      0
FareBin_Code     0
Embarked_Code    0
dtype: int64
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	IsAlone	AgeBin	Title
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	2	0	(16.0, 32.0]	Mr
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	2	0	(32.0, 48.0]	Mrs
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	1	1	(16.0, 32.0]	Miss
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	2	0	(32.0, 48.0]	Mrs
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1	1	(32.0, 48.0]	Mr
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	13.0000	S	1	1	(16.0, 32.0]	Misc
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	30.0000	S	1	1	(16.0, 32.0]	Miss
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.0	1	2	23.4500	S	4	0	(16.0, 32.0]	Miss
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	30.0000	C	1	1	(16.0, 32.0]	Mr
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	7.7500	Q	1	1	(16.0, 32.0]	Mr

Pclass	SibSp	Parch	Age	Fare	FamilySize	IsAlone	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S	Title_Master	Title_Misc	Title_Miss	Title_Mr	Title
3	1	0	22.0	7.2500	2	0	0	1	0	0	1	0	0	0	1	
1	1	0	38.0	71.2833	2	0	1	0	1	0	0	0	0	0	0	
3	0	0	26.0	7.9250	1	1	1	0	0	0	1	0	0	1	0	
1	1	0	35.0	53.1000	2	0	1	0	0	0	1	0	0	0	0	
3	0	0	35.0	8.0500	1	1	0	1	0	0	1	0	0	0	1	

Null Value??

- Age -> Median
- Fare -> Median
- Embarked -> Mode

With or Without Data Pre-processing

Which Model is the BEST?

Let's try as much model as I can ~

p.s. sklearn is a good

Linear Regression

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.3435	0.074	18.268	0.000	1.199	1.488
Sex[T.male]	-0.5056	0.028	-17.924	0.000	-0.561	-0.450
Embarked[T.Q]	-0.0046	0.055	-0.083	0.934	-0.113	0.104
Embarked[T.S]	-0.0632	0.034	-1.836	0.067	-0.131	0.004
Pclass	-0.1723	0.020	-8.509	0.000	-0.212	-0.133
Age	-0.0058	0.001	-5.376	0.000	-0.008	-0.004
SibSp	-0.0415	0.013	-3.174	0.002	-0.067	-0.016
Parch	-0.0155	0.018	-0.853	0.394	-0.051	0.020
Fare	0.0003	0.000	0.891	0.373	-0.000	0.001

Logistic Regression

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
X, y = p_data.loc[:, p_data.columns != "Survived" ].to_numpy(), p_data[["Survived"]].to_numpy().ravel()

clf = LogisticRegression(random_state=0).fit(X, y)
clf.predict(X[:2, :])

clf.predict_proba(X[:2, :])


clf.score(X, y)
```

0.7946127946127947

0.8338945005611672

Naïve Bayes

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Complement Naive Bayes
- Bernoulli Naive Bayes
- … many types

Thomas Bayes
1702 - 1761

Good Afternoon

Naïve Bayes (Before)

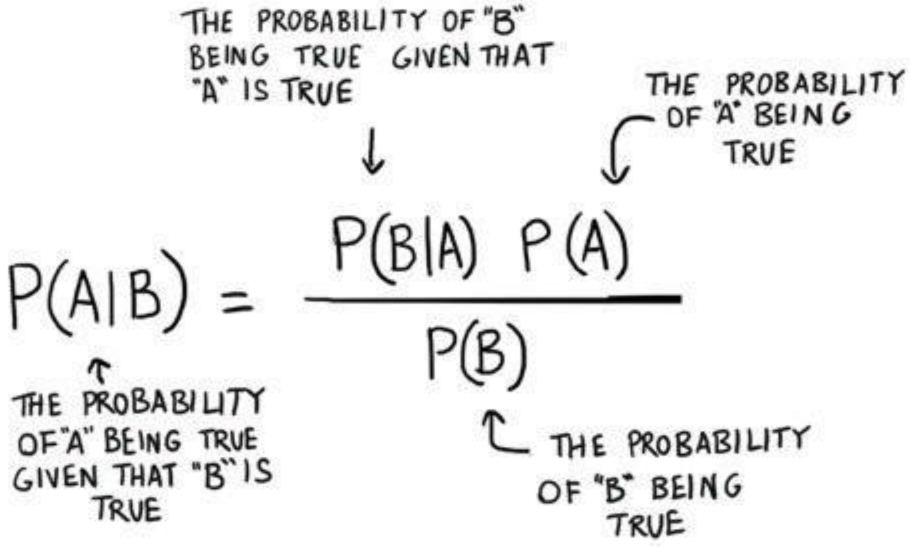
```
nb = GaussianNB()
y_pred = nb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test.to_numpy().flatten() != y_pred).sum()))

nb = MultinomialNB()
y_pred = nb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test.to_numpy().flatten() != y_pred).sum()))

nb = ComplementNB()
y_pred = nb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test.to_numpy().flatten() != y_pred).sum()))

nb = BernoulliNB()
y_pred = nb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test.to_numpy().flatten() != y_pred).sum()))
```

```
Number of mislabeled points out of a total 179 points : 50
Number of mislabeled points out of a total 179 points : 64
Number of mislabeled points out of a total 179 points : 65
Number of mislabeled points out of a total 179 points : 42
```

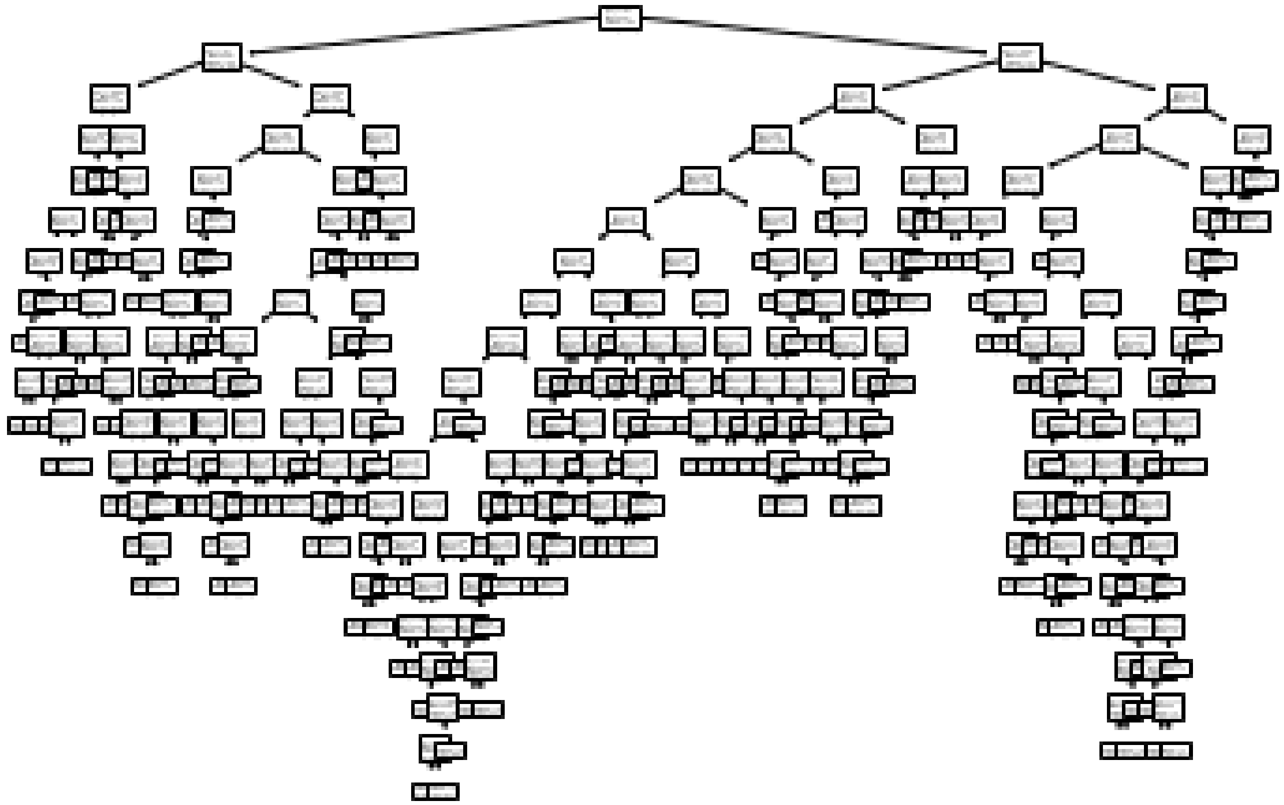


After some data preprocessing..

```
Number of mislabeled points out of a total 179 points : 36
Number of mislabeled points out of a total 179 points : 65
Number of mislabeled points out of a total 179 points : 65
Number of mislabeled points out of a total 179 points : 37
```

Decision Tree

- Overfit?



Tree

- Decision Tree: 76.659%
- Random Forest: 80.002%
- Extra Tree: 79.572%

```python
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier

X, y = p_data.loc[:, p_data.columns != "Survived" ], p_data[["Survived"]]

clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,
    random_state=0)
scores = cross_val_score(clf, X, y, cv=5)
print("Decision Tree:",scores.mean())


clf = RandomForestClassifier(n_estimators=10, max_depth=N
    min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, y, cv=5)
print("Random Forest:",scores.mean())


clf = ExtraTreesClassifier(n_estimators=10, max_dep
    min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, y, cv=5)
print("Extra Tree:",scores.mean())
```

```
Decision Tree: 0.7665934341849224
Random Forest: 0.8002385286548239
Extra Tree: 0.7957253154227606
```

Tree-based feature selection

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectFromModel
X, y = p_data.loc[:, p_data.columns != "Survived" ], p_data[["Survived"]]

print(X.shape)
print(X.columns)
clf = ExtraTreesClassifier(n_estimators=50)
clf = clf.fit(X, y)
print(clf.feature_importances_  )


model = SelectFromModel(clf, prefit=True)
X_new = model.transform(X)
X_new.shape
```

```
(891, 10)
Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 0, 'C', 'Q', 'S'], dtype='object')
[0.10485449 0.28681426 0.23173835 0.04673315 0.05187046 0.24686542
 0.         0.01446903 0.00610297 0.01055185]
```

Error :(

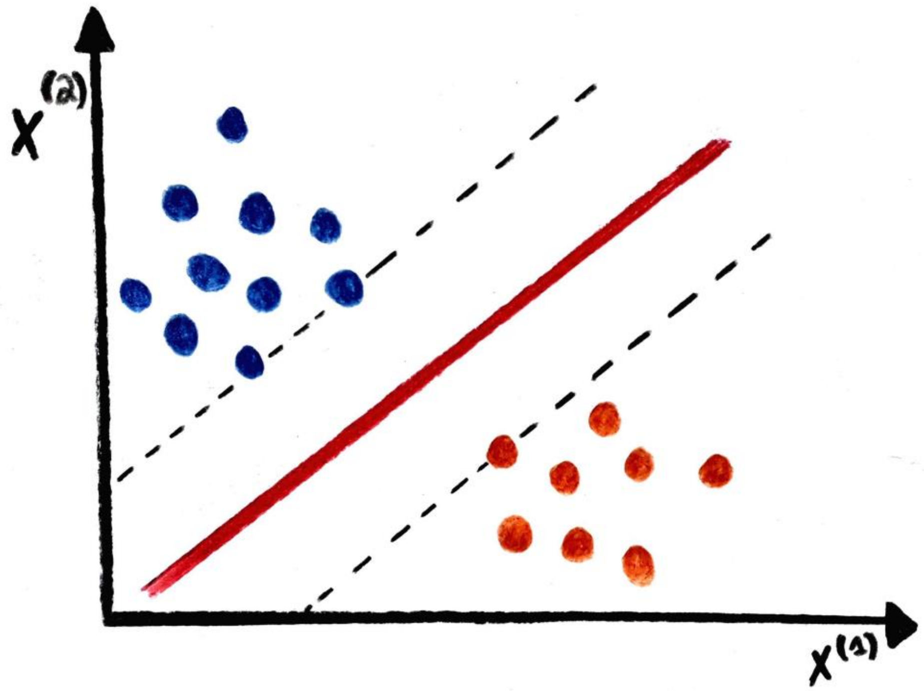
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-56-fc029946c8e8> in <module>
      1 get_ipython().system('pip install graphviz')
----> 2 import graphviz
      3 dot_data = tree.export_graphviz(clf, out_file=No
      4 graph = graphviz.Source(dot_data)
      5 graph.render("iris")

ModuleNotFoundError: No module named 'graphviz'
```

Fixed!!

```
CalledProcessError: Command '['dot', '-Tpdf', '-O', 'iris']' returned non-zero exit status 1. [stderr: b'Form
at: "pdf" not recognized. Use one of:\r\n']
```

Support Vector Machine (SVM)



SVM

```
from sklearn import svm

X, y = p_data.loc[:, p_data.c

kf = KFold(n_splits=5)
clf = svm.SVC()
for train_indices, test_indic
    clf.fit(X[train_indices],
    print("Score:",clf.score(
```

```
Score: 0.6201117318435754
Score: 0.6910112359550562
Score: 0.6741573033707865
Score: 0.6685393258426966
Score: 0.7078651685393258
```

Nearest Neighbours

```
k =  5 : 0.657051282051282
k =  7 : 0.6426282051282052
k =  9 : 0.657051282051282
k =  11 : 0.6634615384615384
k =  13 : 0.6826923076923077
k =  15 : 0.6858974358974359
k =  17 : 0.6875
k =  19 : 0.6858974358974359
k =  21 : 0.6794871794871795
k =  23 : 0.6698717948717948
k =  25 : 0.6778846153846154
k =  27 : 0.6826923076923077
k =  29 : 0.6778846153846154
k =  31 : 0.6778846153846154
k =  33 : 0.6730769230769231
k =  35 : 0.6634615384615384
k =  37 : 0.6810897435897436
k =  39 : 0.6778846153846154
k =  41 : 0.6650641025641025
k =  43 : 0.6762820512820513
k =  45 : 0.6698717948717948
k =  47 : 0.6746794871794872
k =  49 : 0.6634615384615384
```

```
k =  2 : 0.6923076923076923
k =  4 : 0.6730769230769231
k =  6 : 0.6858974358974359
k =  8 : 0.6939102564102564
k =  10 : 0.6971153846153846
k =  12 : 0.7035256410256411
k =  14 : 0.7067307692307693
k =  16 : 0.7051282051282052
k =  18 : 0.6891025641025641
k =  20 : 0.6923076923076923
k =  22 : 0.6971153846153846
k =  24 : 0.6987179487179487
k =  26 : 0.6939102564102564
k =  28 : 0.6907051282051282
k =  30 : 0.6842948717948718
k =  32 : 0.6875
k =  34 : 0.6939102564102564
k =  36 : 0.6858974358974359
k =  38 : 0.6858974358974359
k =  40 : 0.6842948717948718
k =  42 : 0.6875
k =  44 : 0.6858974358974359
k =  46 : 0.6923076923076923
k =  48 : 0.6875
```

MLP

```python
from sklearn.model_selection import KFold
from sklearn.neural_network import MLPClassifier

X, y = p_data.loc[:, p_data.columns != "Survived" ].to_numpy(), p_data[["Survived"]].to_numpy().ravel()

kf = KFold(n_splits=5)
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(3, 3), random_state=1)

for train_indices, test_indices in kf.split(X):
    clf.fit(X[train_indices], y[train_indices])
    print("Score:",clf.score(X[test_indices], y[test_indices]))
```

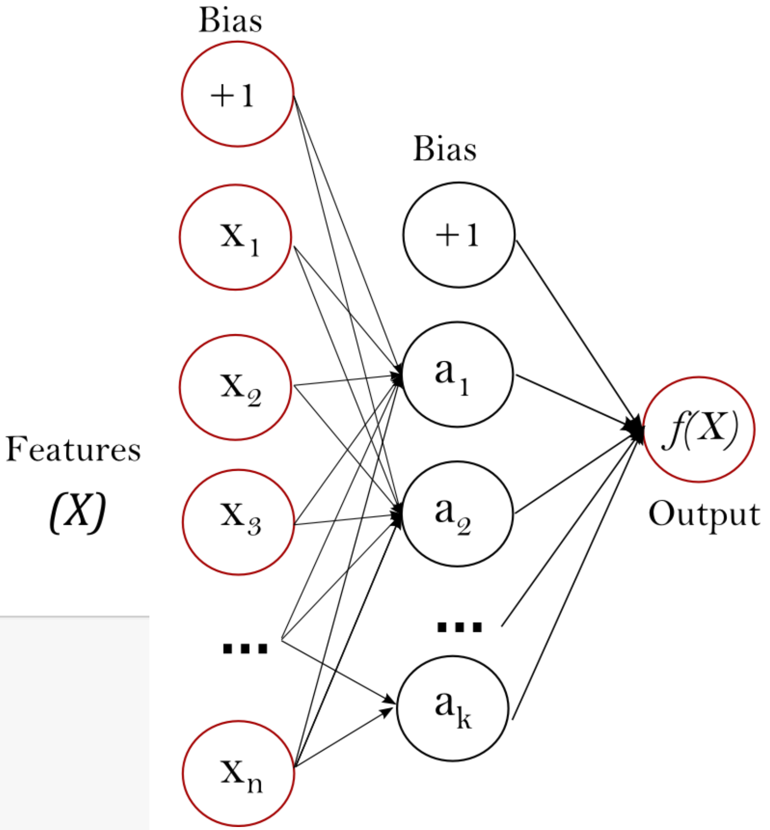
```
Score: 0.7653631284916201
Score: 0.8033707865168539
Score: 0.7640449438202247
Score: 0.8258426966292135
Score: 0.8033707865168539
```

```python
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(7,7,5), random_state=1)

clf.fit(X, y)
cross_val_score(clf, X, y, cv=5)
y_pred_train = clf.predict(X_train)
print("Train Accuracy: ", accuracy_score(y_train, y_pred_train))

y_pred_test = clf.predict(X_test)
print("Valid Accuracy: ", accuracy_score(y_test, y_pred_test))
```

```
Train Accuracy:  0.8342696629213483
Valid Accuracy:  0.8156424581005587
```



Best Model for this problem :)

- Logistic Regression
- Multiple Layer Perceptron