DSP Kaggle Project II

Titanic: Machine Learning from Disaster

Group 2

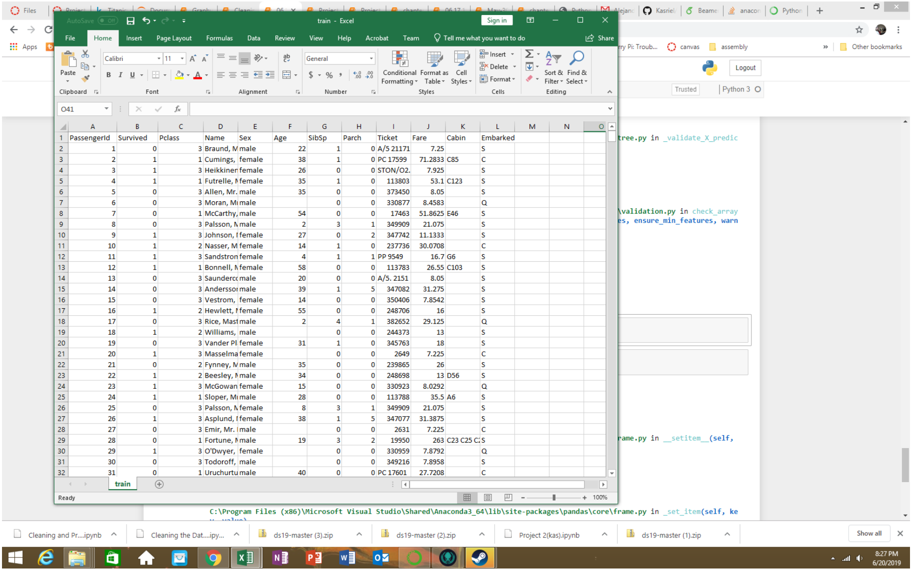
Brian Lai
Kirill Evseev
Alejandra Bermudez

California State University, Northridge

NSF support for the DSP with Career Support and Connections to Industry Program
Award # 1842386

June 20, 2019

Hello Data



PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen	female	26	0	0	STON/O2.	7.925		S
4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q
7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
9	1	3	Johnson, M	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, M	female	14	1	0	237736	30.0708		C
11	1	3	Sandstron	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
13	0	3	Saunderc	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Anderssoı	male	39	1	5	347082	31.275		S
15	0	3	Vestrom,	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, M	female	55	0	0	248706	16		S
17	0	3	Rice, Mast	male	2	4	1	382652	29.125		Q
18	1	2	Williams,	male		0	0	244373	13		S
19	0	3	Vander Pl	female	31	1	0	345763	18		S
20	1	3	Masselma	female		0	0	2649	7.225		C
21	0	2	Fynney, M	male	35	0	0	239865	26		S
22	1	2	Beesley, M	male	34	0	0	248698	13	D56	S
23	1	3	McGowan	female	15	0	0	330923	8.0292		Q
24	1	1	Sloper, Mr	male	28	0	0	113788	35.5	A6	S
25	0	3	Palsson, M	female	8	3	1	349909	21.075		S
26	1	3	Asplund, M	female	38	1	5	347077	31.3875		S
27	0	3	Emir, Mr.	male		0	0	2631	7.225		C
28	0	1	Fortune, M	male	19	3	2	19950	263	C23 C25 C2	S
29	1	3	O'Dwyer,	female		0	0	330959	7.8792		Q
30	0	3	Todoroff,	male		0	0	349216	7.8958		S
31	0	1	Uruchurtu	male	40	0	0	PC 17601	27.7208		C

A look at the Data

In [20]:

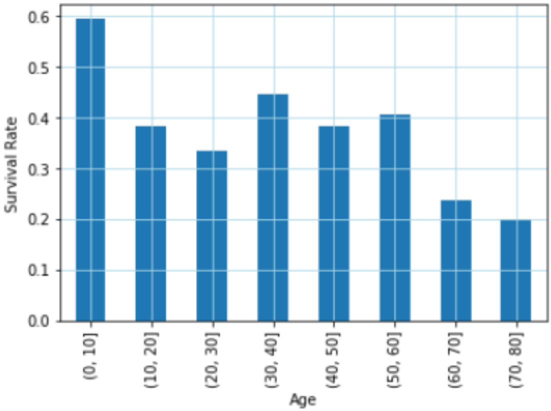
```
# Mean: survival rate
# Count: total observations
# Sum: people survived
# Sex: 0 female, 1 male
data[['Sex', 'Survived']].groupby(['Sex'], as_index=False).agg(['mean', 'count', 'sum'])
```

Out[20]:

	Survived		
	mean	count	sum
Sex			
0	0.754789	261	197
1	0.205298	453	93

Another Look at the Data

```
# survival rate per age
group_by_age = pd.cut(train["Age"], np.arange(0, 90, 10))
age_grouping = train.groupby(group_by_age).mean()
age_grouping['Survived'].plot.bar()
plt.ylabel('Survival Rate')
plt.grid(c="lightblue")
```



How we Cleaned the Data

First Attempt

```
In [12]:
# dropping the data that does not contribute
data = train.drop(columns=['Cabin','Embarked','Name','Ticket','Fare'],axis=0)
```

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

Cleaning the Data

```
# Cleaning the data: first attempt
```

```
In [70]: train3=pd.DataFrame(train)
```

```
In [71]: train3.drop(['Name', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'], axis=1, inplace=True)
```

```
In [72]: train3=train3.dropna(axis=0)
```

```
In [73]: train3['Age']=np.where(train3['Age'].between(0,10),0,train3['Age'])
         train3['Age']=np.where(train3['Age'].between(10.5,30),1,train3['Age'])
         train3['Age']=np.where(train3['Age'].between(30.5,60),2,train3['Age'])
         train3['Age']=np.where(train3['Age'].between(60.5,100),3,train3['Age'])
```

```
In [75]: train3=pd.get_dummies(train3,columns=['Sex'])
         train3=pd.get_dummies(train3,columns=['Age'])
         train3=pd.get_dummies(train3,columns=['Pclass'])
```

Cleaned Data

In [88]:
```
1 train3[:10]
```
Out[88]:

	PassengerId	Survived	Sex_female	Sex_male	Age_0.0	Age_1.0	Age_2.0	Age_3.0	Pclass_1	Pclass_2	Pclass_3
0	1	0	0	1	0	1	0	0	0	0	1
1	2	1	1	0	0	0	1	0	1	0	0
2	3	1	1	0	0	1	0	0	0	0	1
3	4	1	1	0	0	0	1	0	1	0	0
4	5	0	0	1	0	0	1	0	0	0	1
6	7	0	0	1	0	0	1	0	1	0	0
7	8	0	0	1	1	0	0	0	0	0	1
8	9	1	1	0	0	1	0	0	0	0	1
9	10	1	1	0	0	1	0	0	0	1	0
10	11	1	1	0	1	0	0	0	0	0	1

In [30]:
```
tr4=pd.DataFrame(train) #going with tr4 instead of train4
```

In [31]:
```
tr4.drop(['Ticket', 'Fare', 'Cabin', 'Embarked'], axis=1, inplace=True)
```

In [32]:
```
data=[tr4]
titles = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for dataset in data:
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr',\
                                            'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
    dataset['Title'] = dataset['Title'].map(titles)
    dataset['Title'] = dataset['Title'].fillna(0)
tr4 = tr4.drop(['Name'], axis=1)
#taken from https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8
```

In [33]:
```
titleNull=tr4[tr4['Title']==0] #there is nobody with a null title
len(titleNull)
```

Out[33]: 0

Trying to find out the mean age of those who are possibly children or adults with the thinking spouse/sib<1 would mean siblings and 0<parents/children<=2 would probably mean parents if the first was satisfied

In [34]:
```
tr4pc=tr4[tr4['Parch']<=2] #tr4pc is possible child
tr4pc=tr4pc[tr4pc['Parch']>0] #trying no parents when # is in front
tr4pc=tr4pc[tr4pc['SibSp']>1]
#last might mess things up since the assumption is that spouse with no kids but it could be someone with only one sibling
len(tr4pc)
```

Out[34]: 55

In [35]:
```
tr4pa=pd.concat([tr4,tr4pc,tr4pc]).drop_duplicates(keep=False)#tr4pa is possible adult
len(tr4pa)
```

Getting Some Results (GSR)

```
meanAgeMr=mr["Age"].mean()
medAgeMr=mr["Age"].median()
print(meanAgeMr,medAgeMr,len(mr))

miss=tr4LA[tr4LA['Title']==2]
#miss=miss[miss['Parch']==0] #likely older
meanAgeMs=miss["Age"].mean()
medAgeMs=miss["Age"].median()
print(meanAgeMs,medAgeMs,len(miss))

mrs=tr4LA[tr4LA['Title']==3]
meanAgeMrs=mrs["Age"].mean()
medAgeMrs=mrs["Age"].median()
print(meanAgeMrs,medAgeMrs,len(mrs))

rare=tr4LA[tr4LA['Title']==5]
meanAgeRare=rare["Age"].mean()
medAgeRare=rare["Age"].median()
print(meanAgeRare,medAgeRare,len(rare))
```

```
32.56106870229008 30.0 509
27.68617021276596 25.5 121
35.898148148148145 35.0 125
45.54545454545455 48.5 23
```

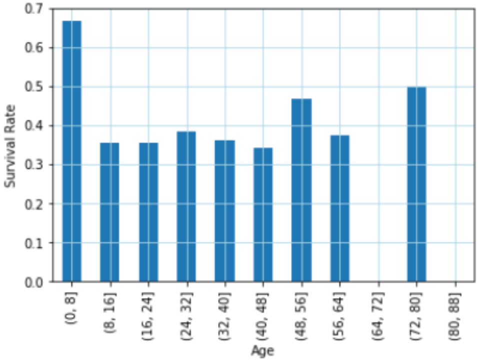
Making an Informed Guess (MIG)

In [45]:
```
#Replace age with mean of each larger group rounded to the near
tr4LCh=tr4LCh.fillna(9.5)
tr4LA=tr4LA.fillna(33)
```

Resulting Age Distribution (RAD)

```
# Survival rate per age of combined
group_by_age = pd.cut(train_combine["Age"], np.arange(0, 90, 8))
age_grouping = train_combine.groupby(group_by_age).mean()
age_grouping['Survived'].plot.bar()
plt.grid(c="lightblue")
plt.ylabel('Survival Rate')
```

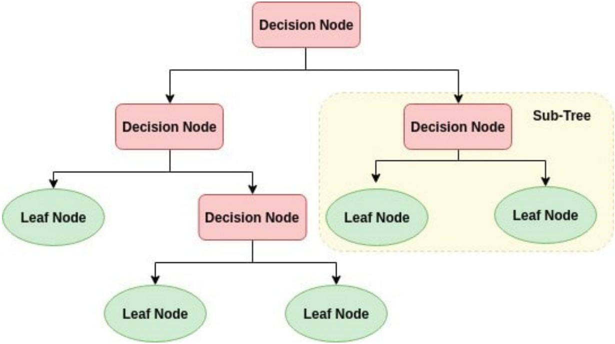
```
Text(0, 0.5, 'Survival Rate')
```



What is our Model?

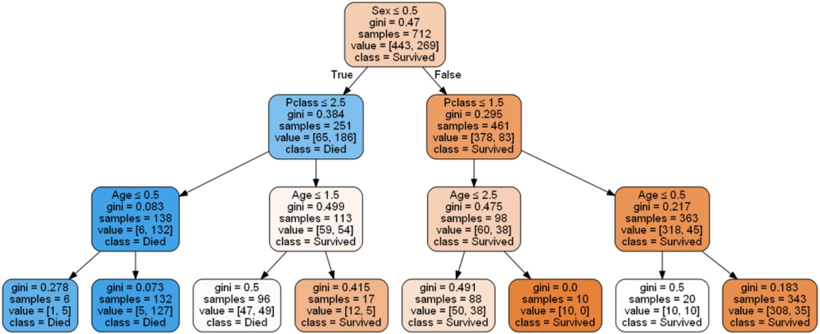
What exactly is a decision tree?

- A Decision Tree is a support tool that uses a flowchart-like model of decisions and potential consequences
- A supervised learning algorithm that works with both continuous and categorical variables
- It is a tool that only contains conditional control statements
- Each leaf node is a class label which is the outcome



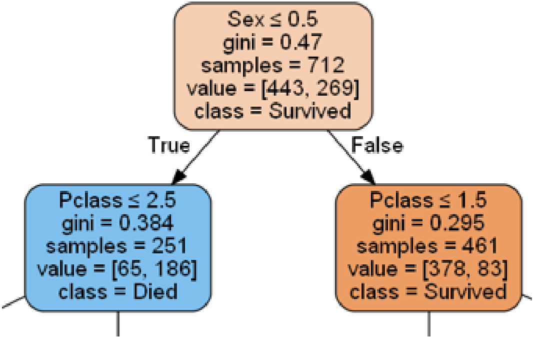
Model: Decision Tree

- Each branch is the outcome of the test
- It's rarely ever balanced
- A depth of 3:
  Sex, Pclass, and Age



Model: Decision Tree

- Most important factor is Sex, it splits off to determine survival/death outcome
- Blue boxes = most likely to die
- Orange boxes = most likely to survive



Fitting the Model

In [78]:
```
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [79]:
```
X=train3.values
Y=train3['Survived'].values
```

In [80]:
```
X=np.delete(X,1,axis=1)
```

In [81]:
```
X=np.delete(X,0,axis=1)
```

In [85]:
```
XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
nsplit=1000
depth=1
```

In [87]:
```
while (depth < 7):
    errs=[]
    for j in range(nsplit):
        XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
        DT=tree.DecisionTreeClassifier(max_depth=depth)
        DT.fit(XTRAIN,YTRAIN)
        YP=DT.predict(XTEST)
        errs.append(1-accuracy_score(YTEST,YP))
    print("Decision Tree Depth = %d mean error = %7.6f SD=%7.6f"\
          %(depth,np.mean(errs),np.std(errs)))
    depth = depth + 1
```

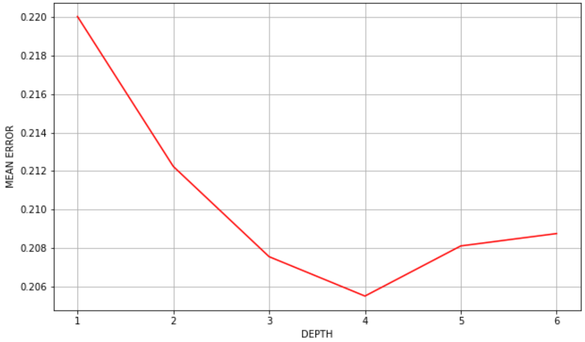
```
Decision Tree Depth = 1 mean error = 0.220464 SD=0.026738
Decision Tree Depth = 2 mean error = 0.213799 SD=0.026424
Decision Tree Depth = 3 mean error = 0.209737 SD=0.026049
Decision Tree Depth = 4 mean error = 0.204296 SD=0.025150
Decision Tree Depth = 5 mean error = 0.209514 SD=0.024734
Decision Tree Depth = 6 mean error = 0.209944 SD=0.024799
```

Determining the Optimal Depth

```
Decision Tree Depth = 1 mean error = 0.220028 SD=0.026047
Decision Tree Depth = 2 mean error = 0.212240 SD=0.027499
Decision Tree Depth = 3 mean error = 0.207553 SD=0.026333
Decision Tree Depth = 4 mean error = 0.205508 SD=0.024900
Decision Tree Depth = 5 mean error = 0.208112 SD=0.025645
Decision Tree Depth = 6 mean error = 0.208754 SD=0.025076
```

In [99]:

```
plt.plot((range(1,7)), errors, color='red')
plt.xlabel("DEPTH")
plt.ylabel("MEAN ERROR")
plt.grid()
plt.gcf().set_size_inches(10,6)
```



Results: Initial Attempt

From our first attempt:
Score $= 0.55980$

Results: Second Attempt

From our second attempt:
Score $= 0.78468$

An increase of 0.22488!

References

https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8?gi=339b274bc177

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

https://blog.patricktriest.com/titanic-machine-learning-in-python/