Getting Started: Titanic Challenge

Brock Moir

University of Victoria

bmoir@uvic.ca

July 15, 2014

Today's Challenge

Goals for Today

- Download and examine Kaggle's Titanic challenge data set
- Learn some NumPy basics
- Build a simple model using scikit-learn's random forest implementation

Advanced Goals for Today

Improve your submission:

- Better feature engineering
- Use other scikit-learn's implementations
- Add more features

Titanic Data

Download the data and some simple models:
`http://www.kaggle.com/c/titanic-gettingStarted`

Training Set:

- Used to build and validate our model
- 891 passengers (rows)
- 10 features (columns)

Test Set:

- Used to make a prediction/submission
- 418 passengers (rows)
- 10 features (columns)

PId	Surv	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
1	0	3	...	male	22	1	0	...	7.25		S
2	1	1	...	female	38	1	0	...	71.2833	C85	C
3	1	3	...	female	26	0	0	...	7.925		S
4	1	1	...	female	35	1	0	...	53.1	C123	S
5	0	3	...	male	35	0	0	...	8.05		S
6	0	3	...	male		0	0	...	8.4583		Q
7	0	1	...	male	54	0	0	...	51.8625	E46	S
8	0	3	...	male	2	3	1	...	21.075		S

- Note that the data set is incomplete

Getting Started with NumPy

Check out the basic model 'gendermodel.py' for some examples

- NumPy is a python library that allows for high level manipulation of multidimensional arrays
- For our purposes it is especially useful for handling rows and columns of data
- Additionally it includes some useful functions such as: sum, size, and mean

accessing two-dimensional NumPy arrays

- indexing: retrieve the element in row m and column n

  array[m,n]

- slicing: retrieve a one dimensional array giving every i^{th} row in column n from m_1 to m_2

  array[m_1:m_2:i,n]

- masking: retrieve a one dimensional array containing only rows in column n where *bool_vec* is True

  array[bool_vec,n]

Getting Started with scikit-learn

Check out my simple example 'mymodel.py'

- scikit-learn is a python library with many useful machine learning algorithms
- It can handle regression, classification, and clustering problems
- It has built in methods for cross-validation

building a random forest

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=100)
forest = forest.fit( Xtrain, Ytrain)
predictions = forest.predict(Xtest)
```

Take a look at the documentation to see what each parameter does:
`http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

Advanced tactics

Some ideas to improve your model

- Change the hyper-parameters of your classifier
- If high bias: add more features (add more columns, make new columns)
- If high varience: add less features (training set is small, model might be over fit)
- Better processing (use a regression model to fill in the blanks)
- Try a different classifier (eg. boosted decision trees)