Using Machine Learning to solve a classification problem with scikit-learn - a practical walkthrough PyConUK 2016

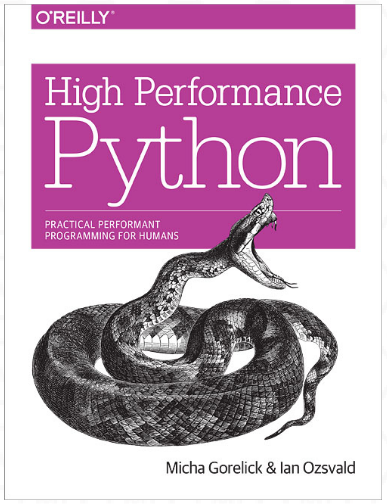
Ian Ozsvald @IanOzsvald ModelInsight.io

Introductions

- I'm an engineering data scientist
- AI/Data Science consulting 15+ years
  - Data science team coach – I observe that engineers have the data

MI ModelInsight

Blog->IanOzsvald.com


O'REILLY
High Performance
Python
PRACTICAL PERFORMANT PROGRAMMING FOR HUMANS
Micha Gorelick & Ian Ozsvald

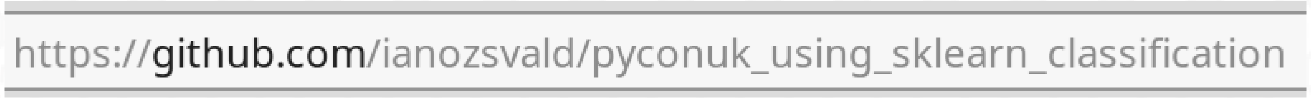
NUMFOCUS
OPEN CODE = BETTER SCIENCE

PyData London

We’ll briefly cover...

- Why? My *hypothesis* about you
- Two class classification
- A process to build an ML model
- Train/Test and Cross validation
- Debugging the model
- Deployment

Fully worked process, more examples, more graphs



https://github.com/ianozsvald/pyconuk_using_sklearn_classification

Ian.Ozsvald@ModelInsight.io @IanOzsvald

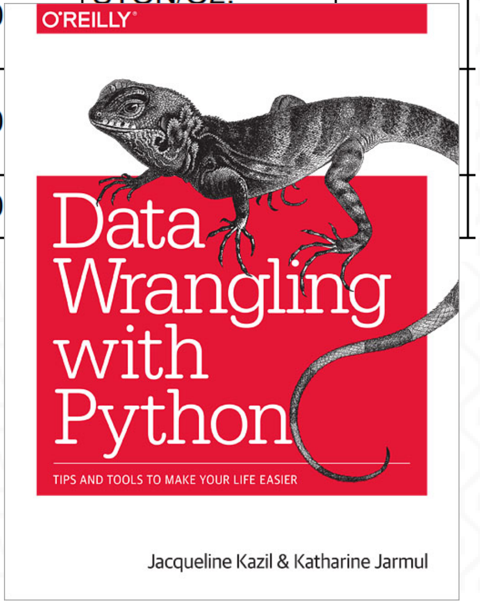
Process

- Exploratory Data Analysis (EDA)
- Build a DummyClassifier model
- Build a RandomForest with several features
- Use cross-validation (Notebook+Appendix) in favour of Train/Test sets
- Find worst errors and improve
- Stop when ‘good enough’ for your needs

Data overview

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2.	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0		
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0		


O'REILLY
Data Wrangling with Python
TIPS AND TOOLS TO MAKE YOUR LIFE EASIER
Jacqueline Kazil & Katharine Jarmul

Knowledge • 4,928 teams

Titanic: Machine Learning from Disaster

Fri 28 Sep 2012 — Sat 31 Dec 2016 (3 months to go)

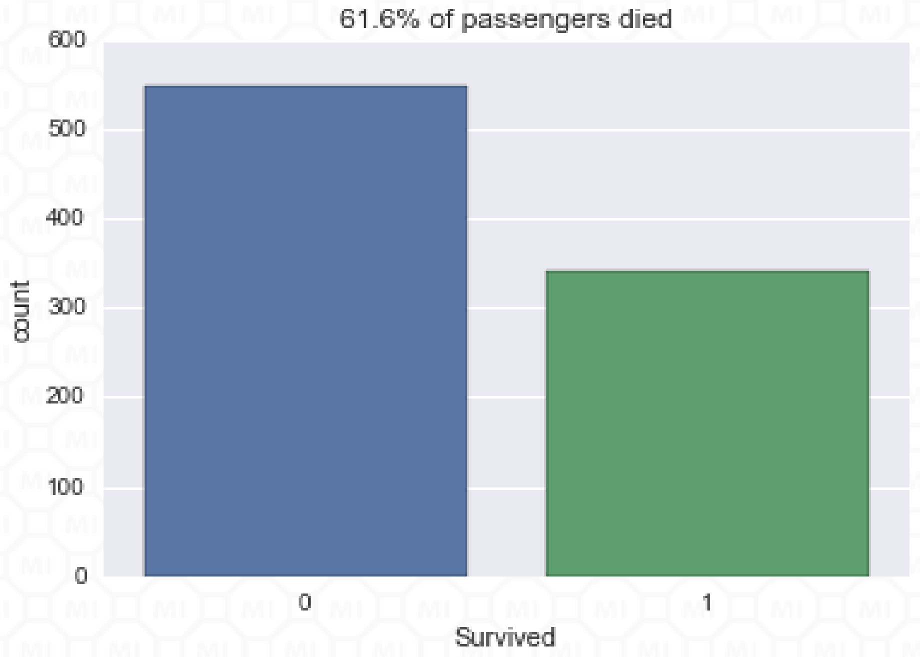
kaggle

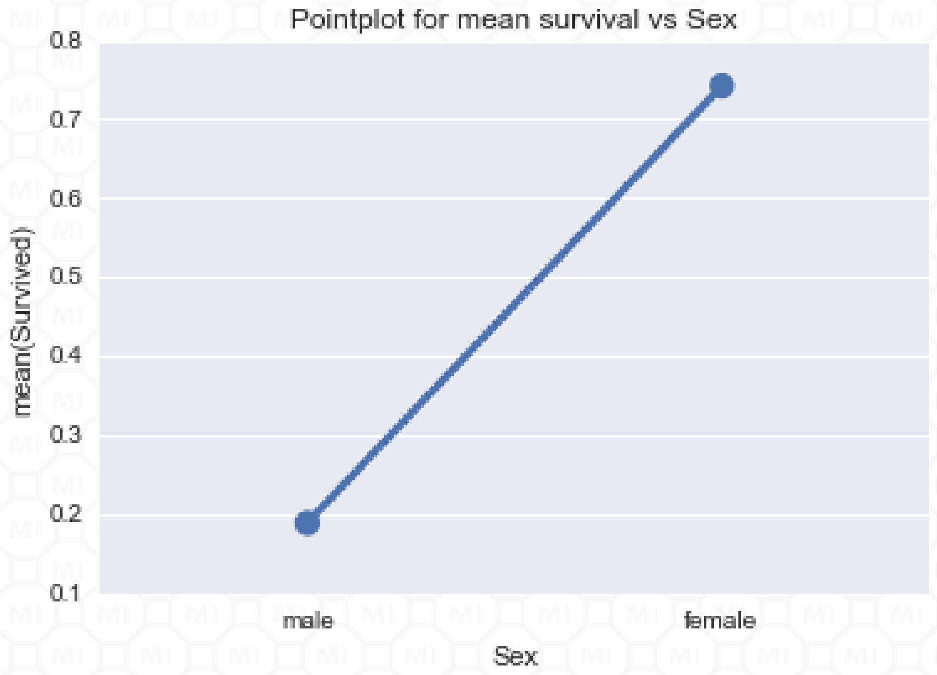
Nice, fairly tidy data – usually you have to work hard here!

Seaborn plots for EDA

```
sns.pointplot(data=df, x='Sex', y='Survived', ci=None)
```

```
_=sns.countplot(df['Survived'])
```



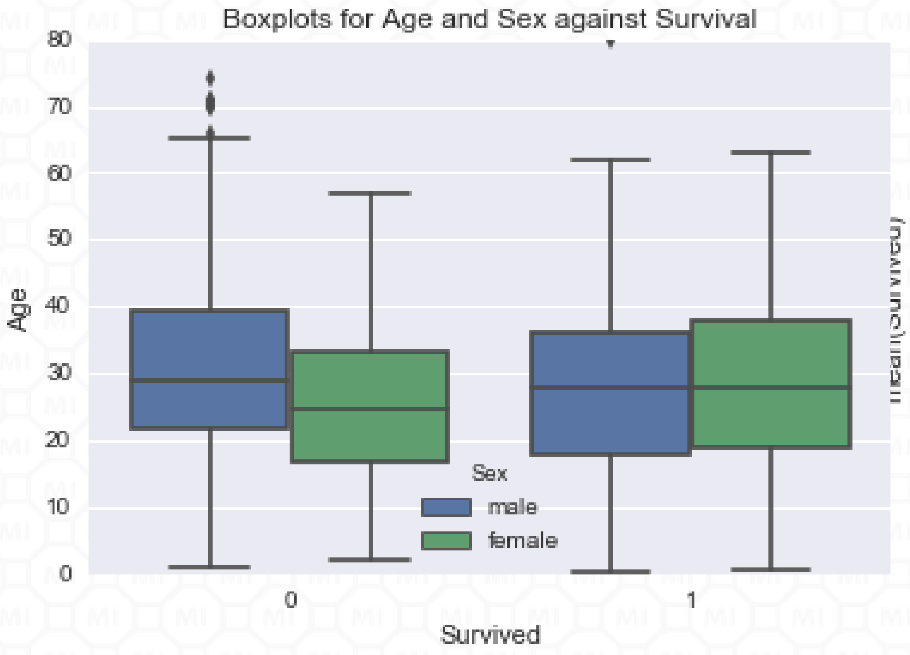


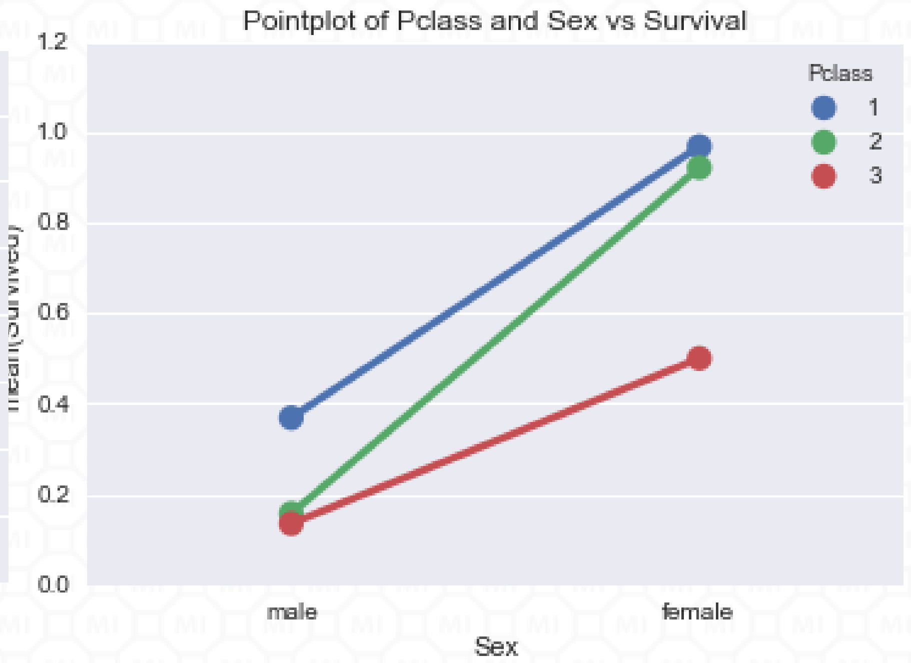
Classifier's best guess is 'they died' unless you introduce new information e.g. 'Sex'

Seaborn plots

```
_=sns.boxplot(data=df, x="Survived", y="Age", hue="Sex")
```

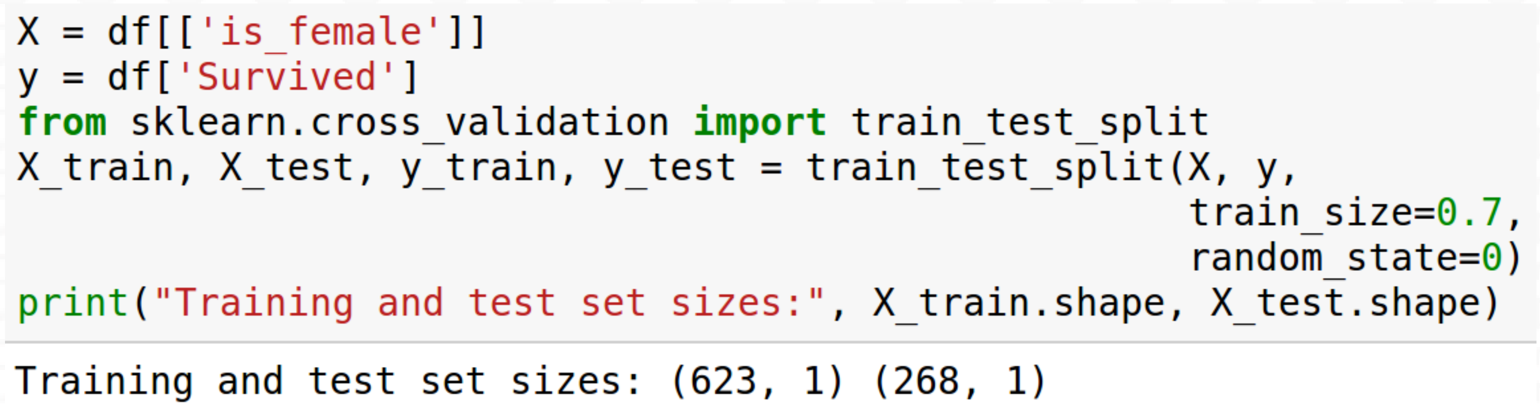
```
sns.pointplot(data=df, x='Sex', y='Survived', hue='Pclass', ci=None)
```





Training and Testing

- Features (X) and Target (y)
- Training and test splits of each
- Like lessons and exams
    - Clever algs can memorize the answers!

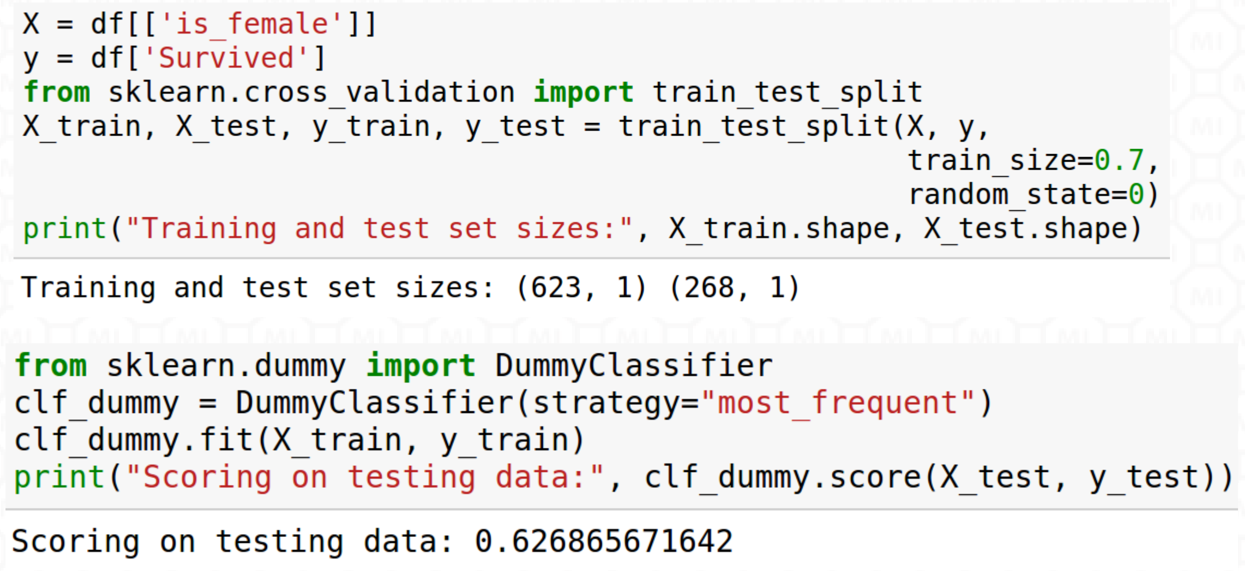


```
X = df[['is_female']]
y = df['Survived']
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.7,
                                                    random_state=0)
print("Training and test set sizes:", X_train.shape, X_test.shape)
```

```
Training and test set sizes: (623, 1) (268, 1)
```

Simplest sklearn

- Do the dumbest thing first – no ML, just a majority-class guess to make a baseline
- ‘Train’ and predict on test set



```
X = df[['is_female']]
y = df['Survived']
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.7,
                                                    random_state=0)
print("Training and test set sizes:", X_train.shape, X_test.shape)
```

```
Training and test set sizes: (623, 1) (268, 1)
```

```
from sklearn.dummy import DummyClassifier
clf_dummy = DummyClassifier(strategy="most_frequent")
clf_dummy.fit(X_train, y_train)
print("Scoring on testing data:", clf_dummy.score(X_test, y_test))
```

```
Scoring on testing data: 0.626865671642
```

Here we ignore is_female, it just makes an appropriately sized input matrix
X ‘stuff to learn’
y ‘target to learn’

Random Forests

- Treat as a ‘black box’
- Very powerful and robust
  - Doesn’t require scaling
  - Handles non-linear responses
  - Handles relationships between parameters
  - Not (too) fooled if you give many noise features

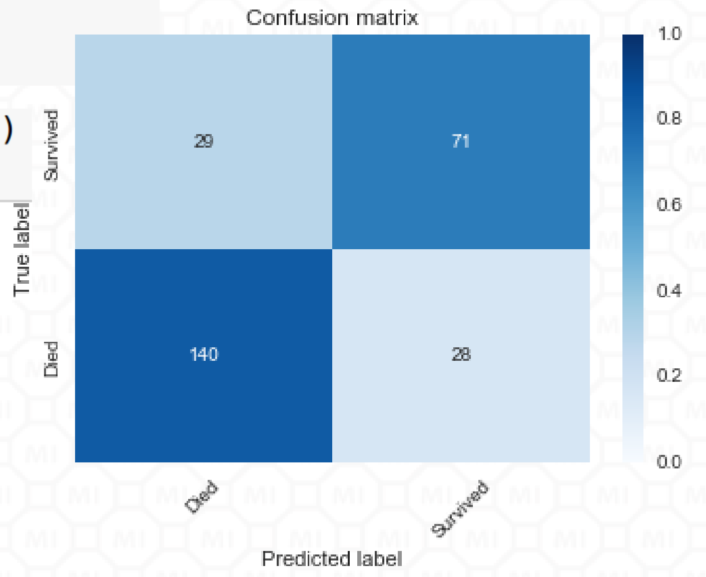
RandomForestClassifier

- Build RF using 1 feature (is_female)
- We outperform a majority guess :-)

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
```

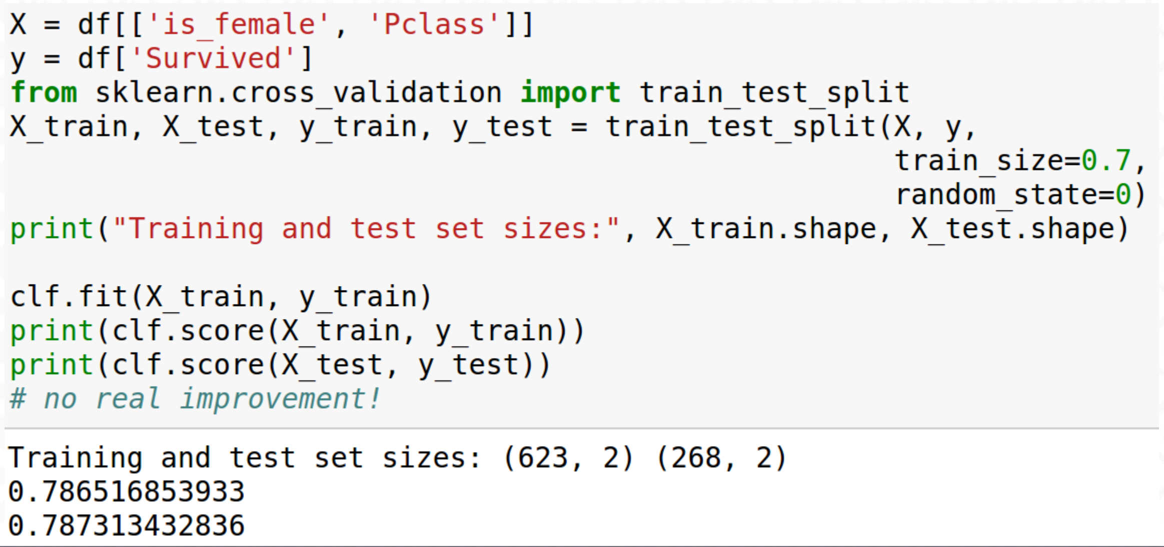
```
print("Scoring on training data:", clf.score(X_train, y_train))
print("Scoring on testing data:", clf.score(X_test, y_test))
```

```
Scoring on training data: 0.786516853933
Scoring on testing data: 0.787313432836
```



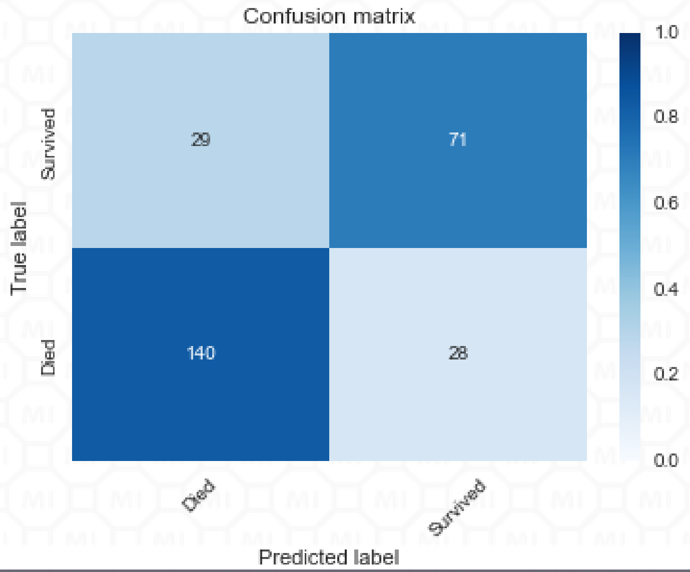
RandomForestClassifier

- Build RF using 2 features
- No significant improvement...we'll push on (this is the usual state…)
  - General rule – add more features



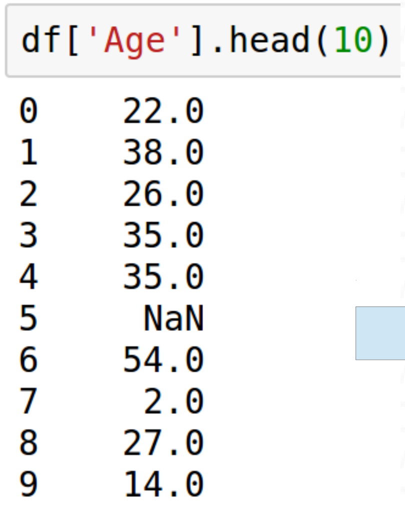
```
X = df[['is_female', 'Pclass']]
y = df['Survived']
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.7,
                                                    random_state=0)
print("Training and test set sizes:", X_train.shape, X_test.shape)

clf.fit(X_train, y_train)
print(clf.score(X_train, y_train))
print(clf.score(X_test, y_test))
# no real improvement!
```

```
Training and test set sizes: (623, 2) (268, 2)
0.786516853933
0.787313432836
```



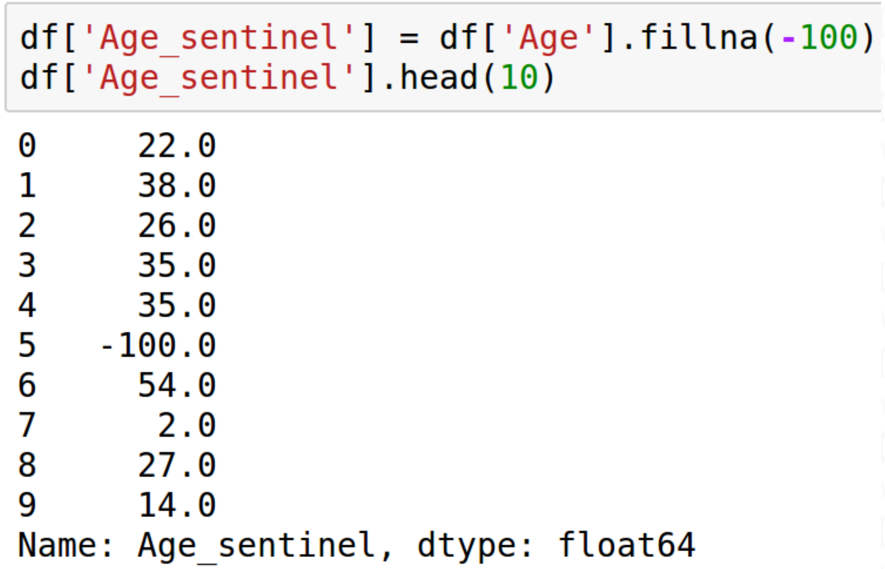
Dealing with NaN/Null

- Sklearn work work with NaN values
- You must replace or delete these rows
- RF works fine if you make a sentinel value



```
df['Age'].head(10)
```

```
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5     NaN
6    54.0
7     2.0
8    27.0
9    14.0
```



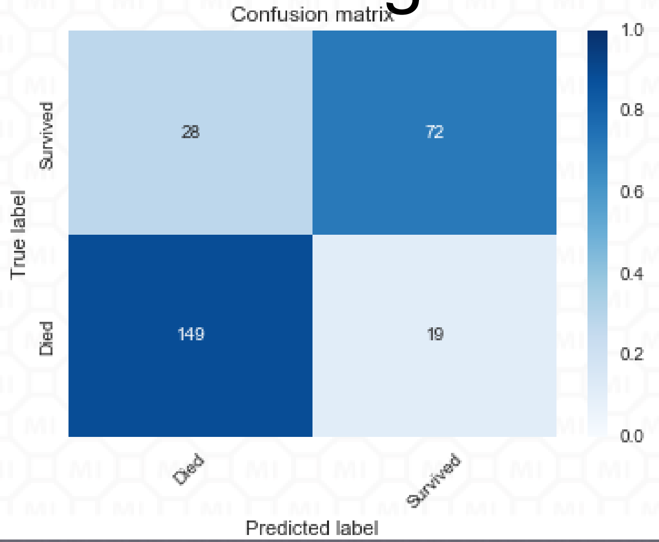
```
df['Age_sentinel'] = df['Age'].fillna(-100)
df['Age_sentinel'].head(10)
```

```
0     22.0
1     38.0
2     26.0
3     35.0
4     35.0
5   -100.0
6     54.0
7      2.0
8     27.0
9     14.0
Name: Age_sentinel, dtype: float64
```

RandomForestClassifier

- Build RF using many features
- With bigger RF we may also classify better
  - n_estimators only param worth tuning

```
clf = RandomForestClassifier(n_estimators=100)

df['is_mr'] = df['Name'].str.count(", Mr.")
df['family_size'] = df['SibSp'] + df['Parch']

# note Age or Pclass by itself isn't so useful, combinec
feature_names = ['is_female', 'Age_sentinel', 'Pclass',
                 'is_mr', 'family_size', 'Fare']
X = df[feature_names]
y = df['Survived']
```

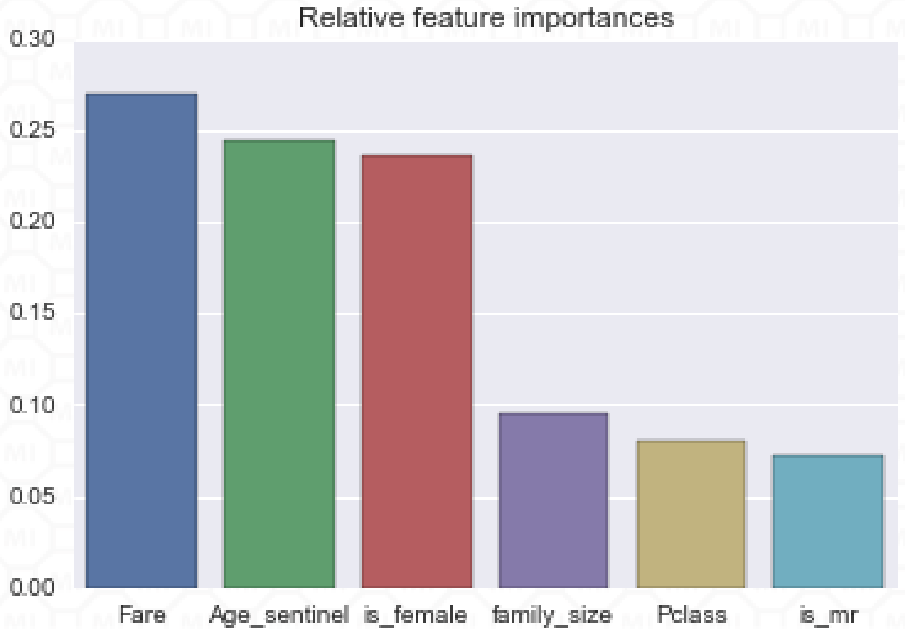


Debugging

- Confusion matrix – does it look sensible?
- Cross validation scores – analogy “many exams” (Notebook+Appendix)
- Feature importances
- Find ‘worst’ errors

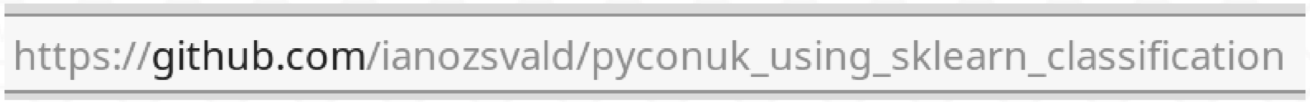
  and eyeball

  (see Notebook)

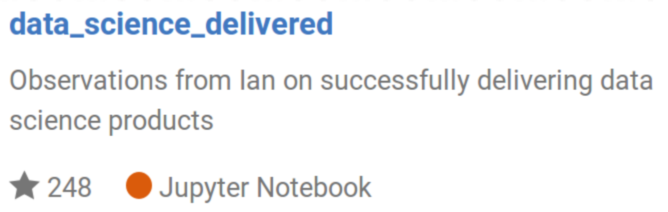


Closing...

- Random Forest + good data gives you a great start
- Write-up: http://ianozsvald.com/
- Use github repo to try this yourself


https://github.com/ianozsvald/pyconuk_using_sklearn_classification

- https://github.com/savarin/pyconuk-introtutorial
  - Longer great tutorial from PyConUK 2014 (Ezzeri)
- Take an engineering mindset and go slow
- Book signing with Katharine!
- Questions<->beer


data_science_delivered

Observations from Ian on successfully delivering data science products

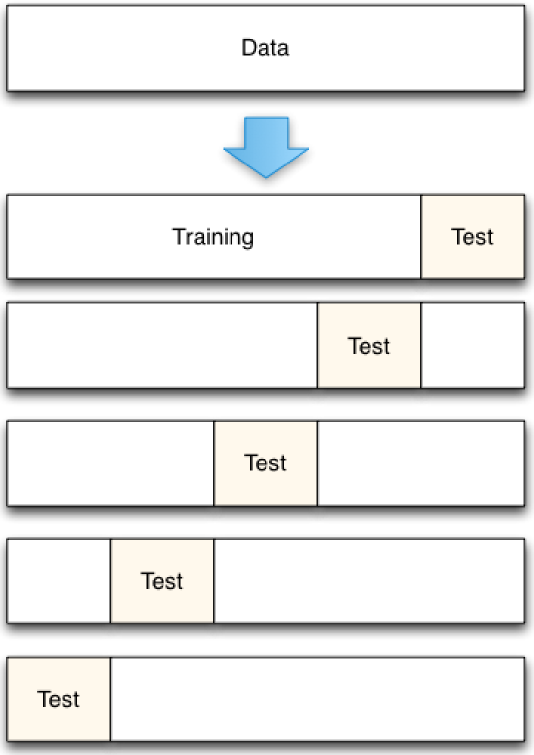
★ 248 ● Jupyter Notebook

Appendix: Deployment

- Pickle your models, reload them
- Ad-hoc scripts → reports or db
- Microservices
  - Flask
  - My featherweight API on github (built on Flask)
  - *New* Jupyter microservices
- Do please have unit tests & reproducible environments
- Use conda environments in Anaconda

Appendix: Cross validation

Sklearn does 3-fold by default (not 5-fold shown here)
3-fold is a sensible starting point
More folds give a better estimate of mean & take longer to run



Ref: http://blog.kaggle.com/2015/06/29/scikit-learn-video-7-optimizing-your-model-with-cross-validation/