Model Evaluation and Validation

Lessons

- https://classroom.udacity.com/nanodegrees/nd101/parts/2a9dba0b-28eb-4b0e-acfa-bdcf35680d90/modules/497059cc-425d-4569-a6a4-c5a71cfaf280/lessons/b13359d2-5f34-4333-9fcd-f5166df104c7/concepts/02585623-835a-4ec2-842d-856326e0450c

---

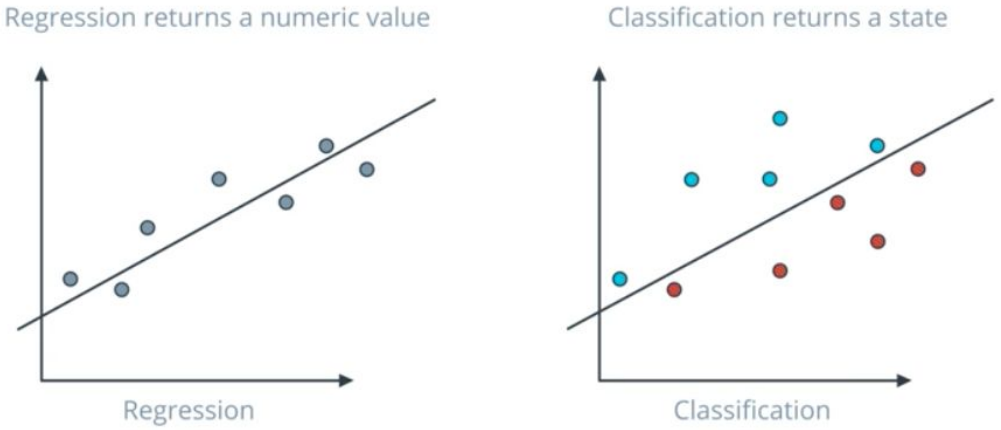
Lesson 1 - Intro

Congratulations! You've built your first model, a neural network to predict bike share usage! As you saw in the project, the data was split into training, validation, and test sets. In this section we'll learn why this is needed, and how to find out how well your model is doing, and score it. In particular, you will cover:

- How to create a test set for your models.
- How to use confusion matrices to evaluate false positives, and false negatives.
- How to measure accuracy and other model metrics.
- How to evaluate regression.
- How to detect whether you are overfitting or underfitting based on the complexity of your model.
- How to use cross validation to ensure your model is generalizable.

Lesson 2 - Testing

- **Regression** is a model that predicts a value (e.g. 4, -3, 9) by fitting a regression line and for any new value, finding the corresponding y value over the line.
- **Classification** is a model where we want to determine a state (e.g. positive/negative, yes/no, cat/dog). We guess a values state based on which 2 regions it belongs.

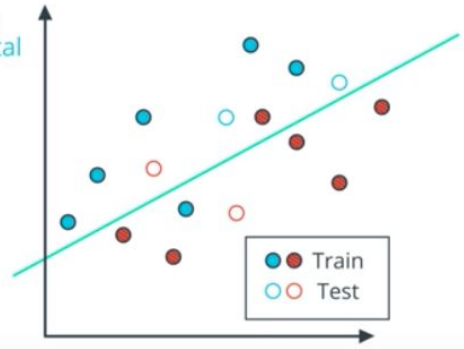


- **Testing** is used to determine which model is better. We split the data into 2 sets:
    - **Training set** is used to train the model.
    - **Testing set** is used to test the model.
- **Choose the model where the errors are small.**

TESTING IN SKLEARN

```
from sklearn.model_selection import train_test_split

X_train, y_train, X_test, y_test =
train_test_split(X,
                 y,
                 test_size = 0.25)
```

4 test
16 total



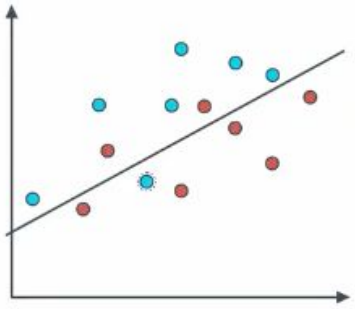
Lesson 3 - Confusion Matrix

- After we develop a model, we want to find out how good it is using a few metrics.
- **Confusion Matrix** is a table that will describe the performance of a model.

SPAM CLASSIFIER MODEL

	Sent to Spam Folder	Sent to Inbox
Spam	True Positive	False Negative
Not Spam	False Positive	True Negative

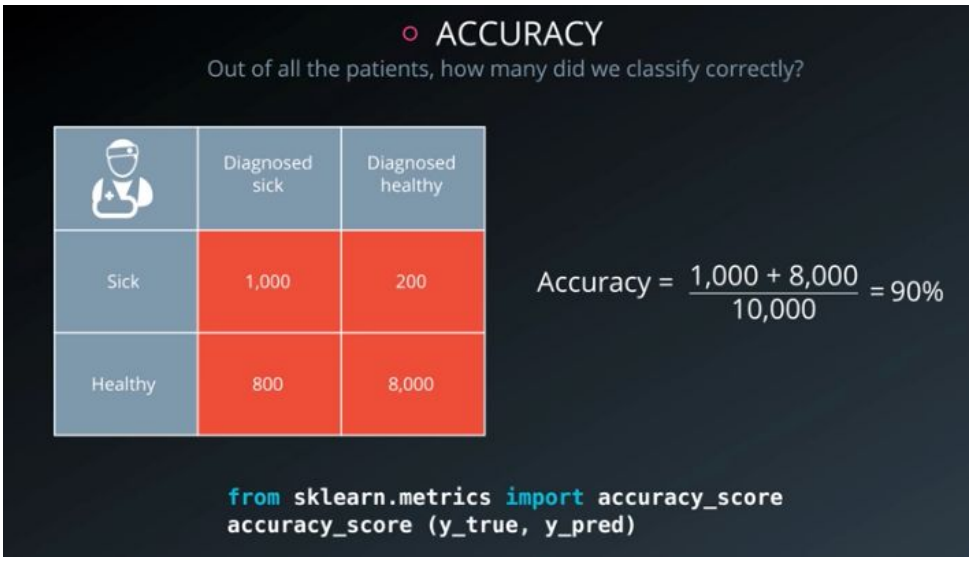
CONFUSION MATRIX



	Guessed Positive	Guessed Negative
Positive	6 True Positives	1 False Negatives
Negative	2 False Positives	5 True Negatives

Lesson 4 - Accuracy

- **Accuracy -** how many points did we classify correctly. $Accuracy = \frac{Number\ of\ correctly\ Classified\ Points}{Total\ number\ of\ points}$

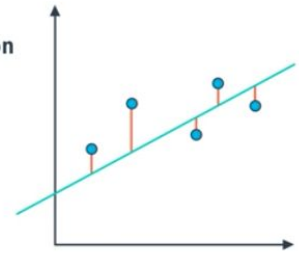


Lesson 5 - Regression Metrics

- **Mean Absolute Error -** add absolute values of distance from points to line.

MEAN ABSOLUTE ERROR IN SKLEARN

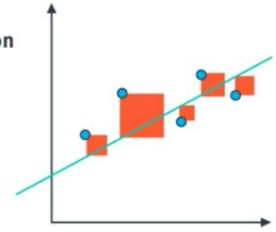
```
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_absolute_error(y, guesses)
```



- **Mean Squared Error -** add squared values of distance from points to line.

MEAN SQUARED ERROR IN SKLEARN

```
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression

classifier = LinearRegression()
classifier.fit(X,y)

guesses = classifier.predict(X)

error = mean_squared_error(y, guesses)
```



- $R2\ Score = 1 - \frac{average\ of\ all\ values}{mean\ squared\ error\ of\ model}$
  - R2 SCORE

BAD MODEL

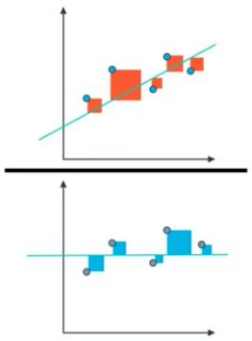
The errors should be similar.
R2 score should be close to 0.

GOOD MODEL

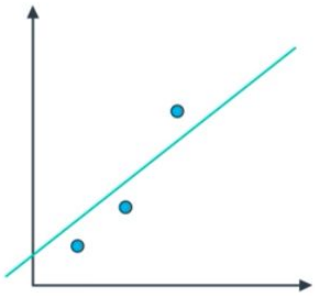
The mean squared error for the linear regression model should be a lot smaller than the mean squared error for the simple model.

R2 score should be close to 1.

R2 = 1 -



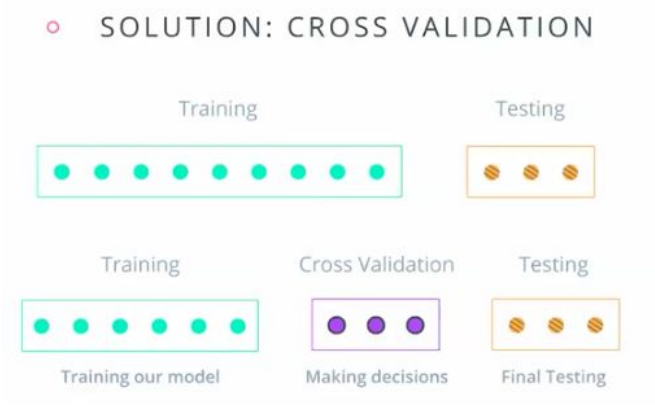
```
from sklearn.metrics import r2_score

y_true = [1, 2, 4]
y_pred = [1.3, 2.5, 3.7]

r2_score(y_true, y_pred)
```

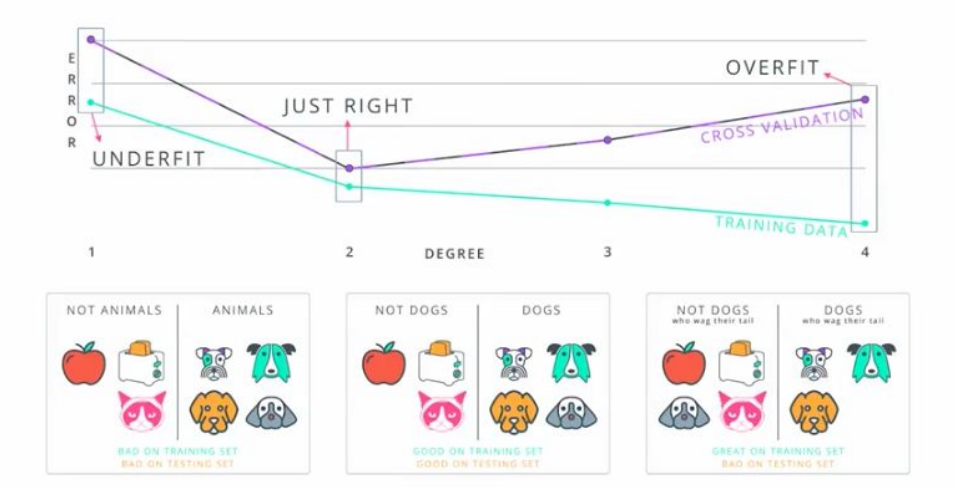


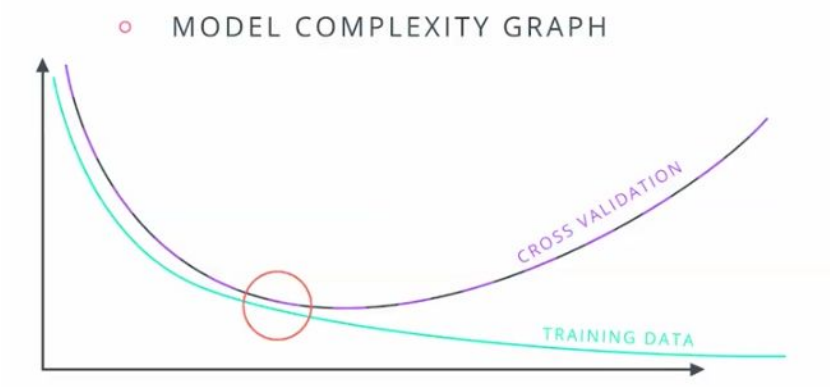
Lesson 6 - Types of Error

- **2 Types of Error:**
  - **Underfitting -** Over-Simplifying problem
    - **Error due to bias** - model doesn't do well in the training set.
  - **Overfitting -** Over-Complicating problem
    - **Error due to variance -** model does well in training set but poorly in testing set.

Lesson 7 - Model Complexity Graph







Lesson 8 - K Fold Cross Validation

- **K Fold Cross Validation -** breaks our data into k buckets and train our model k times, each time using a different bucket as our testing set and the remaining points as our training set. Then we just average the results to get a final model.
    - Randomize data to remove any hint of bias.

CROSS VALIDATION IN SKLEARN

```
from sklearn.model_selection import KFold
kf = KFold(12, 3, shuffle = True)

for train_indices, test_indices in kf:
    print train_indices, test_indices
```

```
[ 0  1  2  3  4  5  6  8  11 ]  [7  9 10]
[ 0  1  2  4  6  7  8  9  10 ]  [3  5 11]
[ 1  3  5  6  7  8  9 10 11]    [0  2  4]
[ 0  2  3  4  5  7  9 10 11]    [1  6  8]
```