Advanced Data Analysis

DATA 71200

Class 3

Schedule

5-Jun	**Inspecting Data**
6-Jun	Async: DataCamp
10-Jun	Evaluation Methods
11-Jun	Async: DataCamp
12-Jun	Supervised Learning (k-Nearest Neighbors, Linear Models, and Naive Bayes Classifiers)
13-Jun	*Async: DataCamp Modules* *Project 1 Due*

Inspecting Data to Gain Insights

- **Review from last class**
    - Data size and type
    - Summary statistics
    - Histograms
    - Scatter Matrix

Representing Data

- **Continuous versus categorical**
  - One-Hot Encoding
  - Binning
- **Transformations**
- **Automatic feature selection**
- **Utilizing expert knowledge**

Some Terminology

- **(Linear) Regression**
    - Continuous predictive model created by estimating a linear relationship between features
- **Logistic Regression**
    - Predictive model of the probability of a certain class

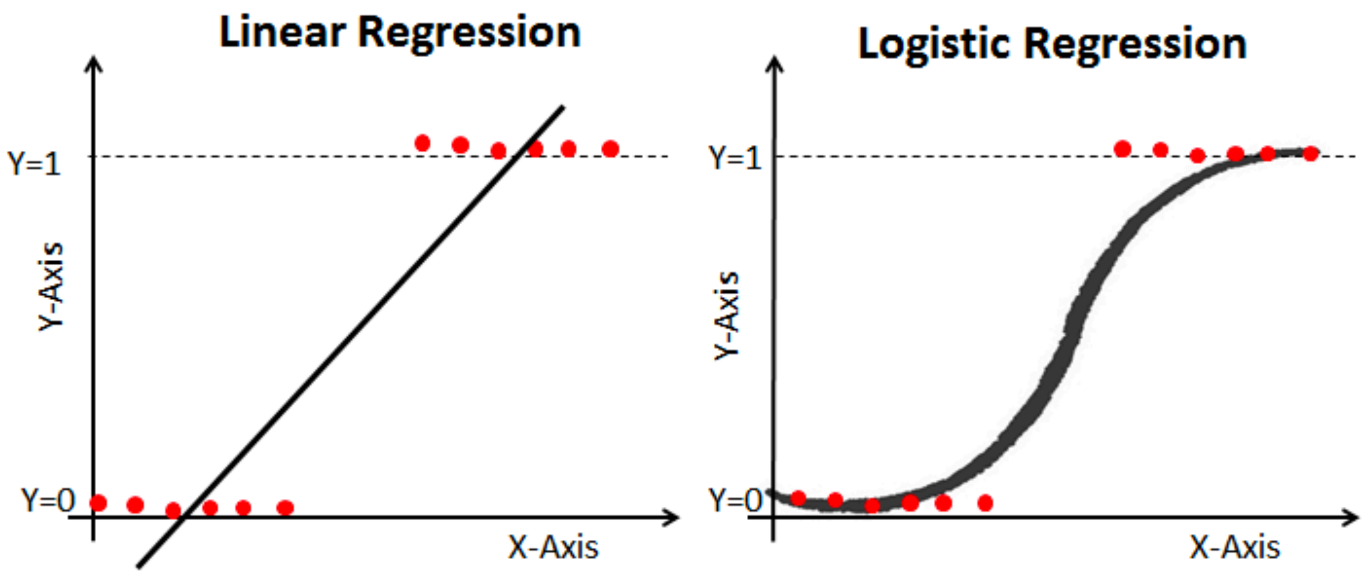

Image: https://medium.com/@hpsuresh12345/logistic-regression-60694a973bee

Some Terminology

- **Regularization**
  - Adds an extra term to the cost function
  - Can be applied to linear and logistic regression
  - Can also be used for feature selection
  - Lasso (least absolute shrinkage and selection operator) regression is another form, referred to as L1
  - Ridge is a form of regularization, referred to at L2

Some Terminology

- **Lasso Regression (L1)**
  - reduces the coefficients of the least important variables to zero (removing them completely by the model)
- **Ridge Regression (L2)**
  - addresses *multicollinearity* (linear relationships between parameters) and having more parameters than observations

Continuous Versus Categorical

- (Linear) Regression - predicts continuous values
- Classification - predicts categorical, or discrete, values
- Continuous versus categorical distinction also holds for input features

One-Hot Encoding

- **Split the different categories in their own variable**
- **E.g., a single variable for color where the values are the strings "blue", "red", "yellow" would be encoded as**

	Blue	Red	Yellow
Blue	1	0	0
Red	0	1	0
Yellow	0	0	1

Variables

Values

Categorical data can also be encoded as numbers

	Categorical Feature	Integer Feature
0	socks	0
1	fox	1
2	socks	2
3	box	1

1 2 3 4 5 6

0 1 2 3 4 5

```
[[0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 1. 0.]
 [0. 0. 1. 0. 0. 1.]
 [1. 0. 0. 0. 1. 0.]]
```

box fox socks 0 1 2

In-Class Activity 1

- **Apply one-hot encoding to the ocean_proximity value in the California Housing dataset that we looked at last class**
  - Using pd.dummies and/or OneHotEncoder from scikitlearn
  - housing[‘ocean_proximity'].values.reshape(-1,1)

Binning

- **Discretizing continuous data into numerical bins can be useful when small differences in value are not significant**
- **E.g., for numerical grade data (out of 100), it may be more useful to give a model how many scores fall into ranges of 5 rather than the continuous data**

82	83	92	93	72	73	87	86	99	97	98	51	52	82	81	87	91	92	61	67

↓

50-54	**55-59**	**60-64**	**65-69**	**70-74**	**75-79**	**80-84**	**85-89**	**90-94**	**95-99**
2	0	1	2	2	0	4	3	4	2

In-Class Activity 2

- **Apply binning to the housing_median_age value in the California Housing dataset that we looked at last class**
  - `housing['housing_median_age'].values.reshape(-1, 1)`
  - Plot both the original data and the binned data
- **Explore binning with other features**

Transformations

- Squaring and cubing is useful for linear regression models
- Logarithms and exponentials are useful for representing your data with a Gaussian distribution, which is useful for mean-based models

In-Class Activity 3

- **Apply the following transformations to housing_median_age in the California Housing dataset that we looked at last class**
    - Squaring (**2)
    - Cubing (**3)
    - np.log
    - np.exp
- **Plot histograms and scatter matrices to explore the resultant data (for **2, **3, and np.log)**

Automatic Feature Selection

- **Regularization can be used to assess the relative importance of features in the performance of a model**
  - Although this can’t tell you anything about features you don’t include
- **Recursive feature elimination (RFE) starts with all features and removes the poorly performing ones**
- **You can also start with one feature and build up a model**

Utilizing Expert Knowledge

- Domain knowledge can be useful for recognizing patterns in data that may be beneficial or detrimental to the model
- This can inform decisions about which features to include and how to represent them

Reading

- Ch 4: "Representing Data/Engineering Features"in Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc. 213–55.

DataCamp for next class

- *Preprocessing for Machine Learning in Python*

Project 1

- **Due June 13**
- **Keep exploring potential datasets**
  - kaggle.com
  - archive.ics.uci.edu/ml/datasets.php
  - libguides.nypl.org/eresources
  - opendata.cityofnewyork.us/data/
- **The data set will need to be labeled as you are going to use it for both supervised and unsupervised learning tasks**
- ***We will go over using IMPUTER to address missing values on Monday***

Project 1

Curation and cleaning of a labeled data set that you will use for the supervised and unsupervised learning tasks in project 2 and 3. The dataset can be built from existing data and should be stored in your GitHub repository.

To **submit** the assignment submit a link to your project Jupyter notebook on Blackboard

The **goal** for this assignment is for you to create a usable dataset from an open-source data collection. You will use your curated dataset for a supervised classification task in Project 2 and an unsupervised learning task in Project 3.

Step 1: Find and download a dataset. Here are some potential places to look

- Amazon's AWS datasets: https://aws.amazon.com/opendata/public-datasets/
- Data Portals: http://dataportals.org/
- Kaggle datasets: http://kaggle.com
- NYPL digitizations: http://libguides.nypl.org/eresources
- NYC Open Data: http://opendata.cityofnewyork.us/data/
- Open Data Monitor: http://opendatamonitor.eu/
- QuandDL: http://quandl.com/
- UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Some guidelines regarding dataset selection:

- Since this data is going to be used for supervised learning, one of its features should be a value that you can predict (e.g., if you chose a real estate dataset and are interested in housing prices, then housing prices for each sample should be available in the dataset)
- Your dataset should be large enough to perform supervised and unsupervised learning on. A useful rule of thumb is that you should have at least 10 samples per dimension or parameter/feature you are fitting.

Step 2: Divide into a training set and a testing set. In a Jupyter notebook, use scikitlearn to divide your data into training and testing sets. Make sure that the testing and training sets are balanced in terms of target classes

```
from sklearn.model_selection import train_test_split

# split data and labels into a training and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, stratify=None)

#a balanced split, percentage of samples for each class, can be obtained with  StratifiedShuffleSplit
#note however that the housing dataset is not a good candidate for this approach
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(X, y):
  X_train = X[train_index]
  X_test = X[test_index]
  y_train = y[train_index]
  y_test = y[test_index]
```

data71200class4.ipynb

Step 3: Explore your training set. In a Jupyter notebook, import your data into a Pandas data frame and use the following pandas functions to explore your data

- DataFrame.info()
- DataFrame.describe()

Step 4: Data cleaning. Address any missing values in your training set. Include the code in your Jupyter notebook and create a second, cleaned, version of your dataset. Then apply the same procedure to your test set (if you are putting in replacement values use IMPUTER in scikitlearn).

- Recall from the *Hands on Machine Learning* book (p. 60) that some options are
  - "Get rid of the corresponding samples."
  - "Get rid of the whole attribute (column)."
  - "Set the values to some value (zero, the mean, the median, etc.)."

Dealing with missing value

data71200class4.ipynb

```python
X = housing.drop(['median_house_value','ocean_proximity'],axis=1)
```

```python
from sklearn.impute import SimpleImputer
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
imp_mean.fit(X)
SimpleImputer()
X = imp_mean.transform(X)
```

```python
# instantiate a model and fit it to the training set
linreg = LinearRegression().fit(X_train, y_train)
```

```python
# instantiate a model and fit it to the training set
linreg = LinearRegression().fit(X_train, y_train)
```

```
Test set score: 0.63
```

```python
# evaluate the model on the test set
print("Test set score: {:.2f}".format(linreg.score(X_test, y_test)))
```

If you want to see which feature has the missing element

```
# you can check each one individually with the following code
housing['longitude'].isnull().values.any()
```

False

```
housing['latitude'].isnull().values.any()
```

False

```
housing['housing_median_age'].isnull().values.any()
```

False

```
housing['total_rooms'].isnull().values.any()
```

False

```
housing['total_bedrooms'].isnull().values.any()
```

True

```
# although this isn't necessary for running Imputer it may be useful to know where exactly the data is missing
# you can also check how many elements are empty
housing['total_bedrooms'].isnull().values.sum()
```

207

```
# or you can generally check if you have any empty elements in your dataframe
#(and thus whether you need to run Imputer)
housing.isnull().values.any()
```

True

data71200class4.ipynb

Step 5: Visualize the data in your training set. At a minimum, use the following pandas functions to visualize the data in your Jupyter notebook.

- DataFrame.hist
- plotting.scatter_matrix()

data71200class2lab.ipynb

Step 6: Apply transformations to your data. In your Jupetyr notebook apply, squaring, cubing, logarithmic, and exponentials transformations to two features in your dataset. Plot the histograms and scatter matrices of the resultant data.

data71200class3lab.ipynb