DSCI 401 Notes

10.18.17

* **Regression** = building a function that fits the data in order to predict a value
* **Regression** and **classification** are **supervised learning**
* Google “Pandas cheat sheet” for help
  + **Import pandas as pd**
  + **Import numpy as np**
  + **Import pprint**
  + **Import matplotlib.pyplot as plt**
  + **Import seaborn as sns**
  + **From sklearn import linear\_model**
  + **From sklearn.preprocessing import PolynomialFeatures**
  + **From sklearn.model\_selection import train\_test\_split**
  + **From sklearn.metrics import r2\_score**
  + **From sklearn.metrics import accuracy\_score** (overall correct rate of prediction)
  + **From sklearn.metrics import precision\_score**
  + **From sklearn.metrics import recall\_score,**
  + **From sklearn.metrics import f1\_score**
  + **From sklearn.metrics import roc\_auc\_score**
  + **From sklearn.metrics import confusion\_matrix**
* cars = **pd.read\_csv**(‘./data/cars.csv’)
* **.head()** returns first 5 rows
* **.describe()** shows statistics about dataframe
* **cars[‘mpg’]** – returns column
* **df[‘col’].mean()** – returns mean
* **.std()** – standard dev
* **.count()** – number of values
* **list(cars)** – returns list of column names
* **cars[ [‘mpg’, ’disp’, ’hp’] ]** – generates new dataframe from selected columns
* **cars[(cars[‘hp’] >= 200) & (cars[‘mpg’] >= 12)]** – filters our cars where horsepower is less than 200 and mpg is less than 12
* **cars.groupby(‘cyl’)[‘mpg’].mean()** – aggregates into groups, selects mpg and gets mean
* **cars.groupby([‘cyl’, ‘am’])[‘mpg’].describe()** – groups into cylinder and transmission type
* **model** **= linear\_model.LinearRegression()** – create new model
* **model.fit(x\_train, y\_train) –** fit model to data
* **mean\_squared\_error(y\_test, preds) –** mean squared error
* **r2\_score(y\_test, preds) –** fit to model, 1 is perfect fit
* **pt1 = sns.lmplot(x=”Angle”, y=”Distance”, data=df, order=1, ci=None, scatter\_kws={‘s’:80}) –** plots linear fit of data
* **pt2 = sns.lmplot(x=”Angle”, y=”Distance”, data=df, order=2, ci=None, scatter\_kws={‘s’:80}) –** quadratic fit of data
* **quad = PolynomialFeatures(degree = 2) –** creates quadratic model
* **data\_x2 = quad.fit\_transform(data\_x) –** creates quadratic fit of data
* **x\_train, x\_test, y\_train, y\_test = train\_test\_split(data\_x, data\_y, test\_size = .2, random\_state=4)** – sets training data and test data
* **linear\_mod = linear\_model.LinearRegression()** – create empty model
* **linear\_mod.fit(x\_train, y\_train) –** fit model to data
* **preds = linear\_mod.predict(x\_test) –** generate predictions
* **print(str(r2\_score(y\_test, preds)) –** print r2 score
* **features = list(df) –** column names
* **del df[‘State’] –** remove column
* **features.remove(‘Bush%’) –** remove y variable from features
* **ddf2 = pd.get\_dummies(df, columns = [‘Region’])**
* **lasso\_mod = linear\_model.Lasso(alpha = a, normalize=True, fit\_intercept=True)**
* **mod = linear\_model.LogisticRegression() –** can use C=<val> to regularize
* **pred\_probs = mod.predict\_proba(x\_test)**
* **prob\_pos = pred\_probs.transpose()[1]**
* **prob\_neg = pred\_probs.transpose()[0]**
* **pred\_df = pd.DataFrame({‘Actual’: y\_test, ‘Predicted Class’: preds, ‘P(0)’: prob\_neg, ‘P(1):prob\_pos})**
* **accuracy\_score(y\_test, preds)**
* **precision\_score(y\_test, preds) –** call it 1, it really is 1
* **recall\_score(y\_test, preds) –** failing to call 1’s actually 1’s
* **f1\_score(y\_test, preds)) –** combine precision and recall
* **roc\_auc\_score(y\_test, preds) –** area under the curve, true positives to false positives, 1 is best obv
* **confusion\_matrix(y\_test, preds) –** diagonal down is where all cases should be

**Build function to test alpha values**

**Build function to import and sort csv**

**Assignment 2**

**Lasso method: test alphas on test data then use on training data model**