EMORY UNIVERSITY

Emory Continuing Education

Predictive Analytics with Python (Regression)

Sridhar Palle, Ph.D.

Consort Institute

Python – Data Science

- Python: Core python, nuts and bolts
- Data Wrangling: NumPy pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
- Visualization: matplotlib Seaborn
- Machine Learning: scikits learn

Data Science - Procedure

- Data Collection
  - Importing, gathering Data or Data Sets

- Data Exploration
  - Examine the data set
    - Visualizations
    - Correlations, statistics

- Data Preparation
  - Remove variables of non-importance
  - Remove outliers, clean-up
  - Normalization (do this after the step below)
  - Remove Missing Values (do this after the step below)

- Train/Test split

- Performance across models
  - Different ML-models (using default hyper parameters)
  - Tune hyper parameters

- Test Models
  - Cross-Validation with k-fold splits
  - Report Average k-fold test scores

Linear Regression

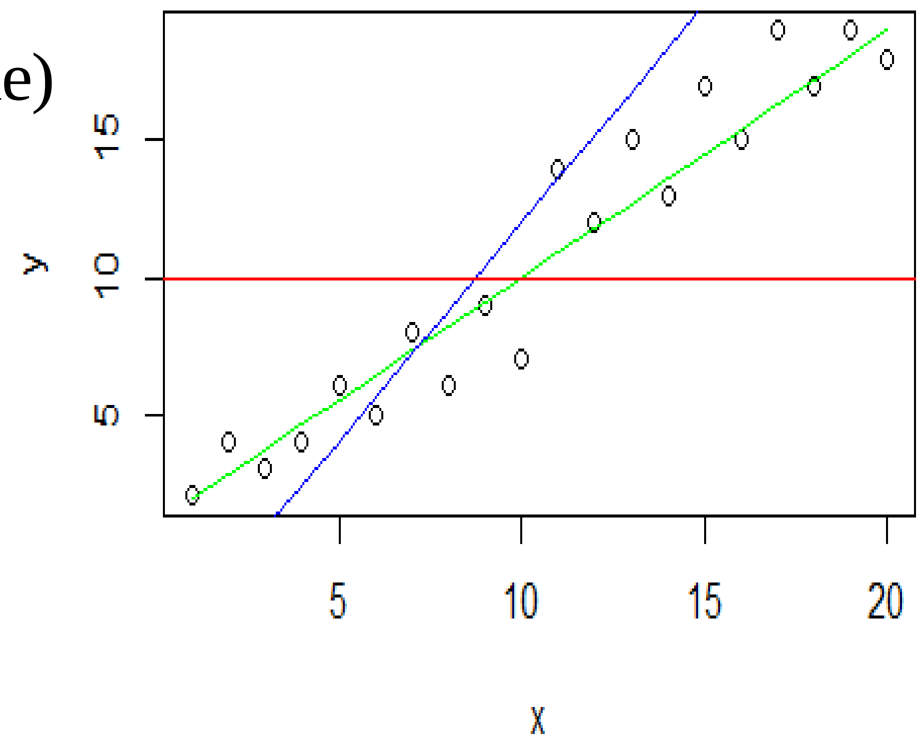
Y - Actual Value of Target Variable

$$\hat{Y} = \beta_0 + \beta_1 x \quad \text{(Predicted Value)}$$

$$SSE = \sum\left(\hat{Y} - Y\right)^2$$

$$SST = \sum\left(\bar{Y} - Y\right)^2$$

$$R^2 = 1 - \frac{SSE}{SST}$$



R2 = 0 (Implies no improvement over base line model)
R2 = 1 (Perfect Model and Fit)

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N-1)}{N-p-1}$$

Algorithm

1. Initialize β_0, β_1

Loop over some iterations or until min SSE{

2. Compute SSE over all examples $SSE = \sum \left(\beta_0 + \beta_1 x - Y \right)^2$

3. Minimize SSE on β_0, β_1
   - Gradient Descent

$$\beta_0 = \beta_0 - \alpha \frac{\partial SSE}{\partial \beta_0} \qquad \beta_1 = \beta_1 - \alpha \frac{\partial SSE}{\partial \beta_1}$$

4. Repeat step 2

Source: Andrew Ng

Scikit-Learn Basics

1. Load Data Sets

a) Built-in datsets

from sklearn import datasets

dir(datasets) – will list the available data sets

dset = datasets.load_diabetes() – this is a dictionary which contains actual data, target variable, feature names, and description

b). Other datasets **can be loaded in as Numpy arrays or data frames**

pd.read_csv(), pd.read_table(), pd.read_excel()

2. Data Preparation

Train/test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Standardization

```
from sklearn.preprocessing import StandardScaler
scaler  = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Polynomial Features

```
from sklearn.preprocessing import PolynomialFeatures
poly =  PolynomialFeatures()
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.fit_transform(X_test)
```

3. Training a model

Few ML algorithms

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Training & making predictions (Ex: linear regression)

```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

y_pred = lr.predict(X_test) – for predictions on test data

y_prob = lr.predict_proba(X_test) – for probabilities on test data

4. Evaluation

a) Regression

```
from sklearn.metrics import r2_score, mean_squared_error

r2_score(y_test,y_pred)
mean_squared_error(y_test,y_pred)
```

b) Classification

```
from sklearn.metrics import accuracy_score, recall_score
        ,precision_score, confusion_matrix, roc_auc_score

confusion_matrix(y_test,y_pred)
accuracy_score(y_test,y_pred)
recall_score(y_test,y_pred)
precision_score(y_test,y_pred)
roc_auc_score(y_test,y_prob)
```