DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 23 2021

Announcements

- Homework 3 will be out later today
  - Due on Monday, March 8
- Project proposal is due on March 4
- Midterm exam is on Tuesday, March 2
  - During class on Gradescope, 11:45am:1:45pm
  - Short review on Thursday, Feb. 25

Project Proposal

- Project Title
- Project Team
- Problem Description
  - What is the prediction problem you are trying to solve?
- Dataset
  - Link to data, brief description, number of records, feature dimensionality (at least 20K records)
- Approach and methodology
  - Normalization
  - Feature selection
  - Machine learning models you will try (recommended >= 4)
  - Language and packages you plan to use
- Metrics (how you will evaluate your models)
- References
  - How did you find out about the dataset, did anyone else used the data for a similar prediction task

Outline

- Generative vs Discriminative Models
- Linear Discriminant Analysis (LDA)
  - Training and inference
  - Why LDA is a linear classifier
- Lab Logistic Regression, LDA, and kNN
- Density Estimation and Naïve Bayes

Generative vs Discriminative

- Generative model
  - Given X and Y, learns the joint probability $P(X, Y)$
  - Can generate more examples from distribution
  - Examples: LDA, Naïve Bayes, language models (GPT-2, GPT-3, BERT)

- Discriminative model
  - Given X and Y, learns a decision function for classification
  - Examples: logistic regression, kNN

LDA

- Classify to one of k classes
- Logistic regression computes directly
  - $\mathrm{P}[Y = 1|X = x]$
  - Assume sigmoid function
- LDA uses Bayes Theorem to estimate it
  - $\mathrm{P}[Y = k|X = x] = \frac{\mathrm{P}[X = x|Y = k]\mathrm{P}[Y=k]}{\mathrm{P}[X=x]}$
  - Let $\pi_k = \mathrm{P}[Y = k]$ be the prior probability of class k and $f_k(x) = \mathrm{P}[X = x|Y = k]$

LDA

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

Continuous Random Variables

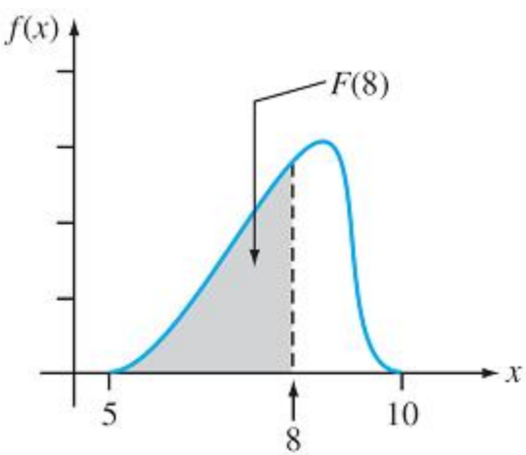
- X:U⟶V is continuous RV if it takes infinite number of values
- The **cumulative distribution function CDF** *F: R* ⟶ {0,1} for *X* is defined for every value *x* by:

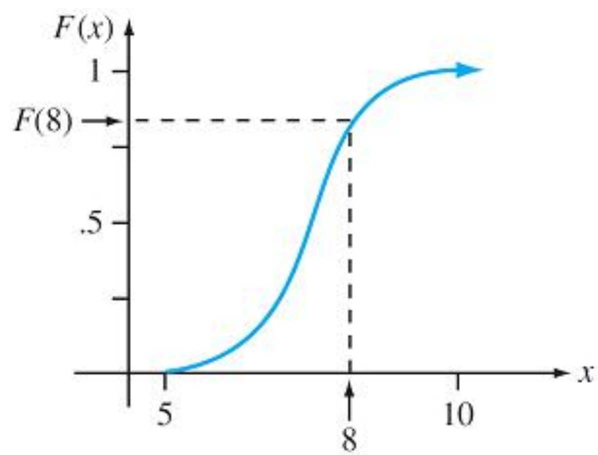
$$F(x) = \Pr(X \leq x)$$

- The **probability distribution function PDF** *f*(*x*) for *X is*

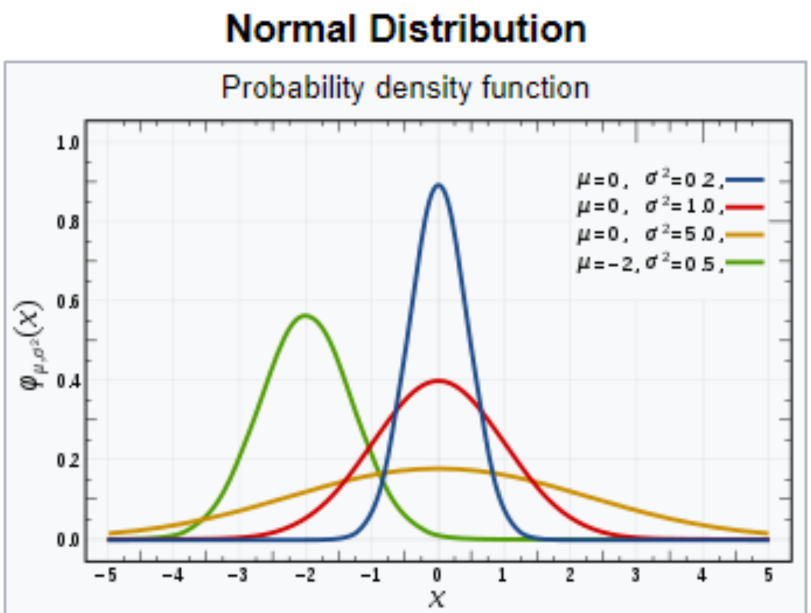
$$f(x) = dF(x)/dx$$

Increasing

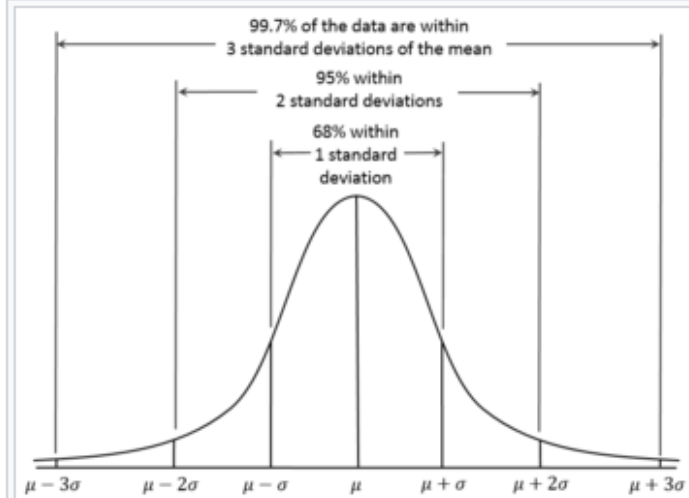




Gaussian Distribution



The red curve is the *standard normal distribution*



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Notation	$\mathcal{N}(\mu, \sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 > 0$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

LDA

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Assumption: $\sigma_1 = \ \ldots \sigma_k = \ \sigma$

LDA Training and Testing

Given training data $(x_i, y_i), i = 1, \dots, n, y_i \in \{1, \dots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k / n.$$

Given testing point x, predict k that maximizes:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}.$$

LDA decision boundary

Pick class k to maximize

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$

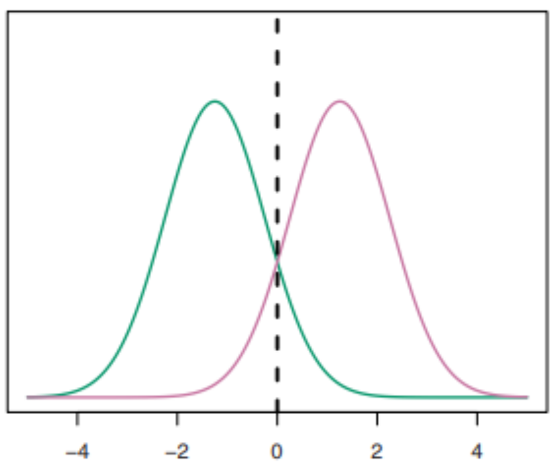
LDA decision boundary

Pick class k to maximize

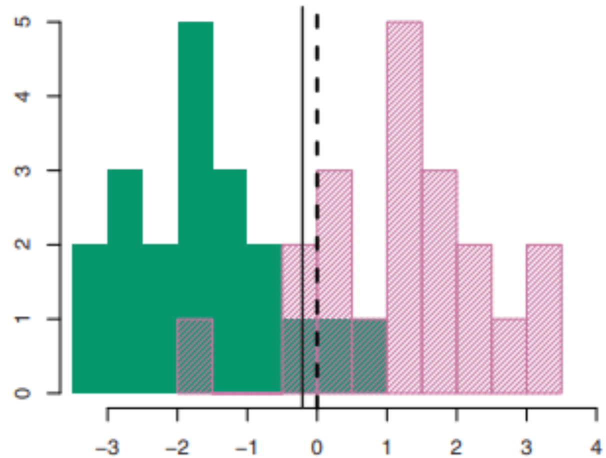
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2, \pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2}$



True decision boundary



Estimated decision boundary

LDA Training and Testing

Given training data $(x_i, y_i), i = 1, \ldots, n, y_i \in \{1, \ldots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

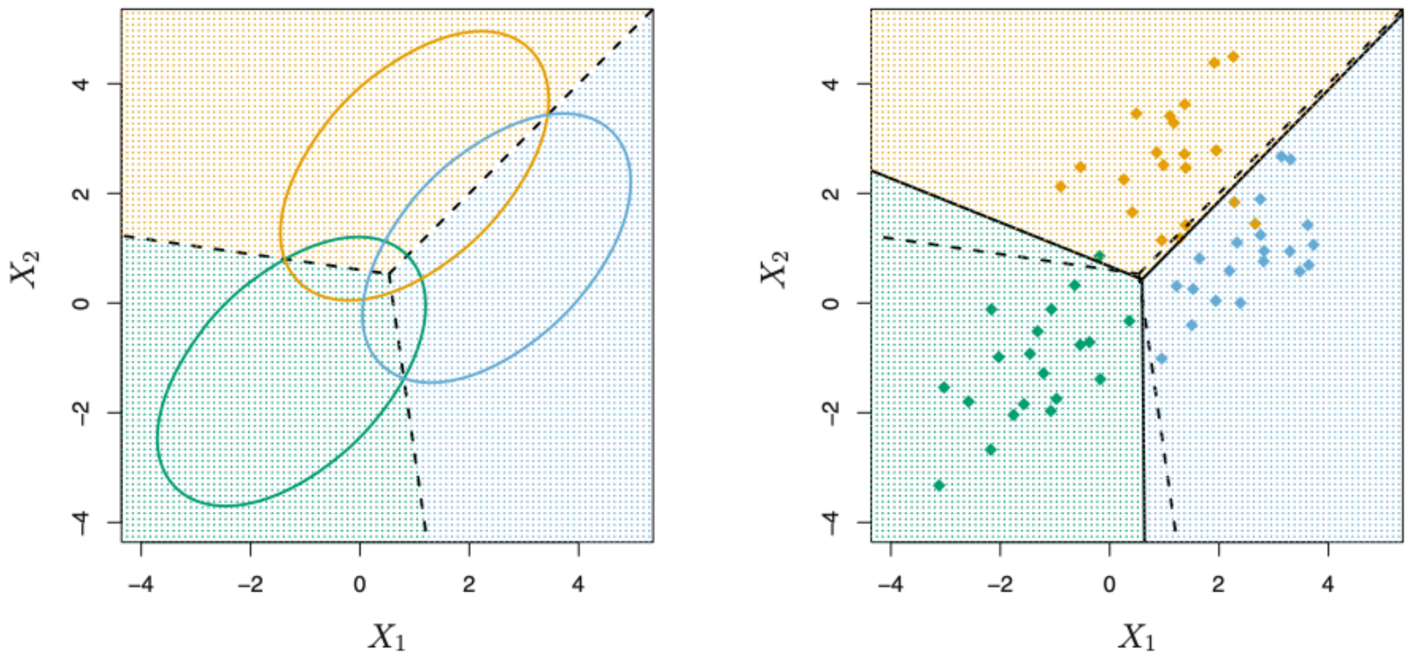
2. Estimate prior

$$\hat{\pi}_k = n_k/n.$$

Given testing point x, predict k that maximizes:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

Multi-Dimensional LDA

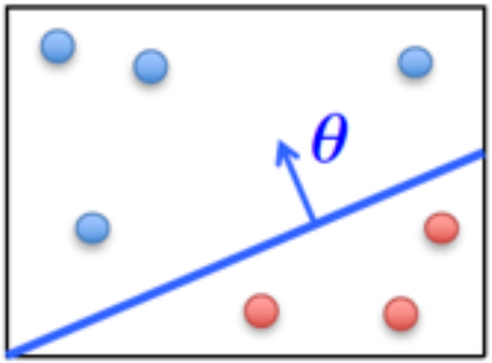


- LDA can be extended to multi-dimensional data
- Assumption that $f_k(x)$ is a multi-variate Gaussian

Linear models

- Logistic regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\top}\boldsymbol{x}}}$$



- LDA

$$Max_k \;\; \delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

LDA vs Logistic Regression

- Logistic regression computes directly $\Pr[Y = 1|X = x]$ by assuming sigmoid function
  - Uses Maximum Likelihood Estimation
  - Discriminative Model
- LDA uses Bayes Theorem to estimate it
  - Estimates mean, co-variance, and prior from training data
  - Generative model
  - Assumes Gaussian distribution for $f_k(x) = \Pr[X = x|Y = k]$
- Which one is better?
  - LDA can be sensitive to outliers
  - LDA works well for Gaussian distribution
  - Logistic regression is more complex to solve, but provides a better model usually

Linear Classifier Lab

```
data = pd.read_csv('heart.csv')
data = data.dropna()
x_columns = data.columns != 'target'
data = utils.shuffle(data)
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
215	43	0	0	132	341	1	0	136	1	3.0	1	0	3	0
145	70	1	1	156	245	0	0	143	0	0.0	2	0	2	1
190	51	0	0	130	305	0	1	142	1	1.2	1	0	3	0
90	48	1	2	124	255	1	1	175	0	0.0	2	2	2	1
166	67	1	0	120	229	0	0	129	1	2.6	1	2	3	0

https://www.kaggle.com/ronitf/heart-disease-uci

Metrics for LR

```
def print_metrics(y_pred, y_true):
    y_pred = np.array(list(map(int, (y_pred > .5))))
    print("TPR: %.2f" % (sum((y_true == 1) & (y_pred == 1)) / sum(y_true == 1)))
    print("FPR: %.2f" % (sum((y_true == 0) & (y_pred == 1)) / sum(y_true == 0)))
    print("TNR: %.2f" % (sum((y_true == 0) & (y_pred == 0)) / sum(y_true == 0)))
    print("FNR: %.2f" % (sum((y_true == 1) & (y_pred == 0)) / sum(y_true == 1)))
```

```
pred_label = logistic_model.predict(x_test)
accuracy = logistic_model.score(x_test, y_test)
error = 1-accuracy
print("Accuracy=",accuracy)
print("Error=",error)

print_metrics(pred_label,y_test)
```

```
Accuracy= 0.8552631578947368
Error= 0.14473684210526316
TPR:  0.98
FPR:  0.30
TNR:  0.70
FNR:  0.02
```

Classification Report

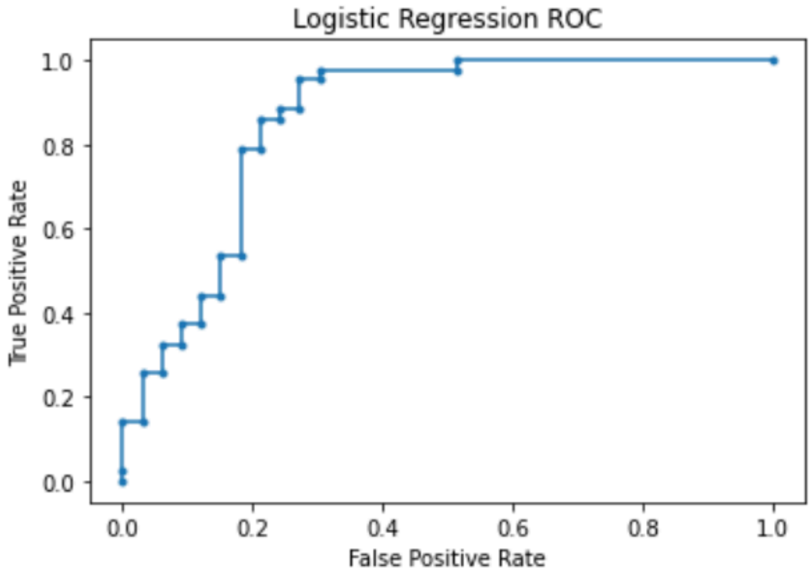
```
from sklearn.metrics import classification_report

target_names = ['class 0', 'class 1']
print(classification_report(y_test, pred_label, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.86      0.79      0.83        39
     class 1       0.80      0.86      0.83        37

    accuracy                           0.83        76
   macro avg       0.83      0.83      0.83        76
weighted avg       0.83      0.83      0.83        76
```

ROC Curve

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

pred_lr = logistic_model.predict_proba(x_test)
pred_lr = pred_lr[:, 1]
r_auc = roc_auc_score(y_test, pred_lr)
print("AUC=",r_auc)

lr_fpr, lr_tpr, _ = roc_curve(y_test, pred_lr)
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('Logistic Regression ROC')
```

```
AUC= 0.8604651162790697

Text(0.5, 1.0, 'Logistic Regression ROC')
```



Lab LDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
print('Priors:')
print(lda.priors_)
print('Means:')
print(lda.means_)
print('Coefficients:')
print(lda.coef_)
print('Test Accuracy:')
print(lda.score(x_test, y_test))
```

```
Priors:
[0.41409692 0.58590308]
Means:
[[5.70744681e+01 8.19148936e-01 4.78723404e-01 1.34882979e+02
  2.49031915e+02 1.27659574e-01 4.36170213e-01 1.40021277e+02
  5.21276596e-01 1.62446809e+00 1.18085106e+00 1.24468085e+00
  2.57446809e+00]
 [5.24060150e+01 5.48872180e-01 1.36090226e+00 1.29548872e+02
  2.45052632e+02 1.27819549e-01 5.93984962e-01 1.59195489e+02
  1.35338346e-01 5.84962406e-01 1.64661654e+00 3.30827068e-01
  2.12030075e+00]]
Coefficients:
[[-5.12655671e-03 -1.65128336e+00  9.42708811e-01 -1.63429905e-02
  -8.26945654e-05  3.61220910e-01  6.53320414e-01  2.61543171e-02
  -1.10225766e+00 -5.26885663e-01  9.83938578e-01 -1.00983532e+00
  -1.16829536e+00]]
Test Accuracy:
0.8026315789473685
```

LDA Metrics

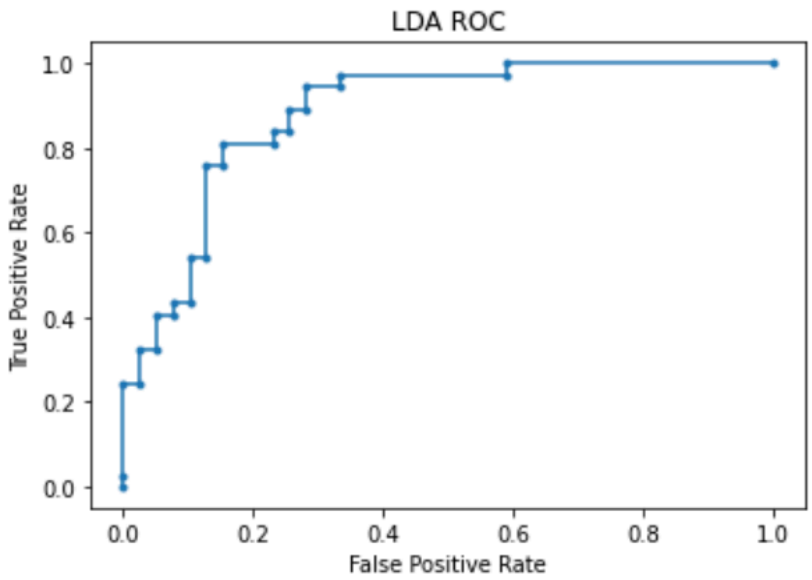
```
target_names = ['class 0', 'class 1']
pred_label_lda = lda.predict(x_test)
print(classification_report(y_test, pred_label_lda, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.88      0.72      0.79        39
     class 1       0.75      0.89      0.81        37

    accuracy                           0.80        76
   macro avg       0.81      0.80      0.80        76
weighted avg       0.81      0.80      0.80        76
```

LDA ROC Curve

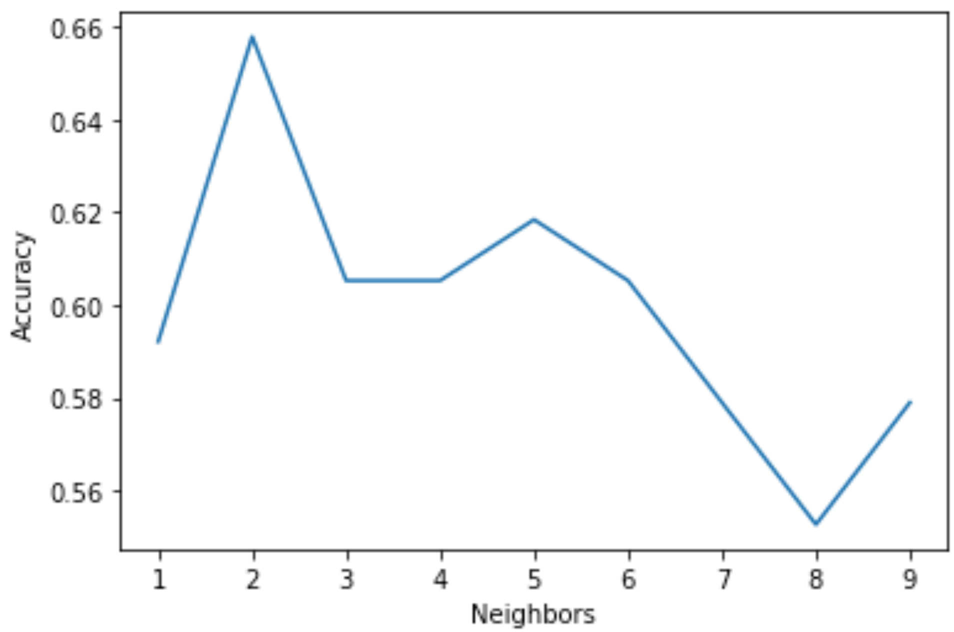
```
pred_lda = lda.predict_proba(x_test)[:,1]
r_auc_lda = roc_auc_score(y_test, pred_lda)
print("AUC=",r_auc_lda)

lda_fpr, lda_tpr, _ = roc_curve(y_test, pred_lda)
pyplot.plot(lda_fpr, lda_tpr, marker='.', label='Logistic')
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.title('LDA ROC')
```

```
AUC= 0.8842688842688843
```



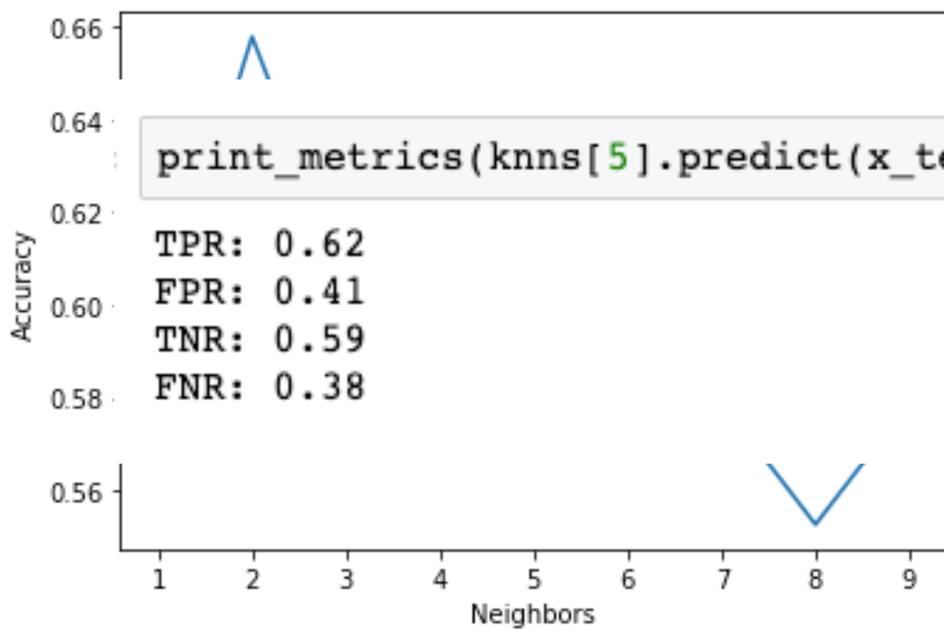
Lab kNN

```python
from sklearn.neighbors import KNeighborsClassifier
accuracies = []
neighbors = list(range(1, 10))
knns = []
for n in neighbors:
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(x_train, y_train)
    knns.append(knn)
    accuracies.append(knn.score(x_test, y_test))
plt.figure().add_subplot(111, xlabel="Neighbors", ylabel="Accuracy")
plt.plot(neighbors, accuracies)
plt.show()
```



Lab kNN

```python
from sklearn.neighbors import KNeighborsClassifier
accuracies = []
neighbors = list(range(1, 10))
knns = []
for n in neighbors:
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(x_train, y_train)
    knns.append(knn)
    accuracies.append(knn.score(x_test, y_test))
plt.figure().add_subplot(111, xlabel="Neighbors", ylabel="Accuracy")
plt.plot(neighbors, accuracies)
plt.show()
```




```python
print_metrics(knns[5].predict(x_test), y_test)
```

```
TPR:  0.62
FPR:  0.41
TNR:  0.59
FNR:  0.38
```

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!