Feature Selection

What are features, anyway?

In machine learning, features represent the transformation of *raw input data* to data that *is formatted* for compatibility with algorithms.

Feature Selection

What are features, anyway?

In machine learning, features represent the transformation of *raw input data* to data that *is formatted* for compatibility with algorithms.

Making a Hypothesis

Part of cleaning up the data involves hypothesizing about the features *you think you'll need* and preparing them to be put through your model.

...If you are a baby crawling on the living room floor, you may want to think about where you want to go.

Assessing the Data

Assessing data means you need to decide what features you think would work well with your model, and it also means you need to determine if you should remove any values that may be missing so the data isn't skewed or biased.

...If you are a baby crawling on the living room floor, you may want to know if there are things in your path. You likely don't need to recall what you had for lunch so you can put that data point to the side.

Selecting Features

data isn't skewed or biased.

...If you are a baby crawling on the living room floor, you may want to know if there are things in your path. You likely don't need to recall what you had for lunch so you can put that data point to the side.

Selecting Features

...You are a baby crawling on the living room floor and you decide to head to your playpen. You'll need your blanket and your bottle, so you grab those.

Your bottle reminds you that ***you are*** hungry, so knowing that you just had mushy peas for lunch might actually be a critical data point that influences what you do next.

Feature Engineering

Feature engineering is the process of creating new features from existing ones. Like feature selection, the process of feature engineering is highly iterative.

...With time and practice babies get quite good at merging similar features into fewer columns in their minds.

1. First, they analyze what stands out and what's easy to find.

2. Then, they create new features. There are a lot of ways to do this, and one is by sorting: "Thing on my plate = I can eat", "thing with hard edges = I could get hurt", "furry thing in the house = I can touch".

The information trail you follow depends on what you're paying attention to, and what you pay attention to *biases the patterns you construct.*

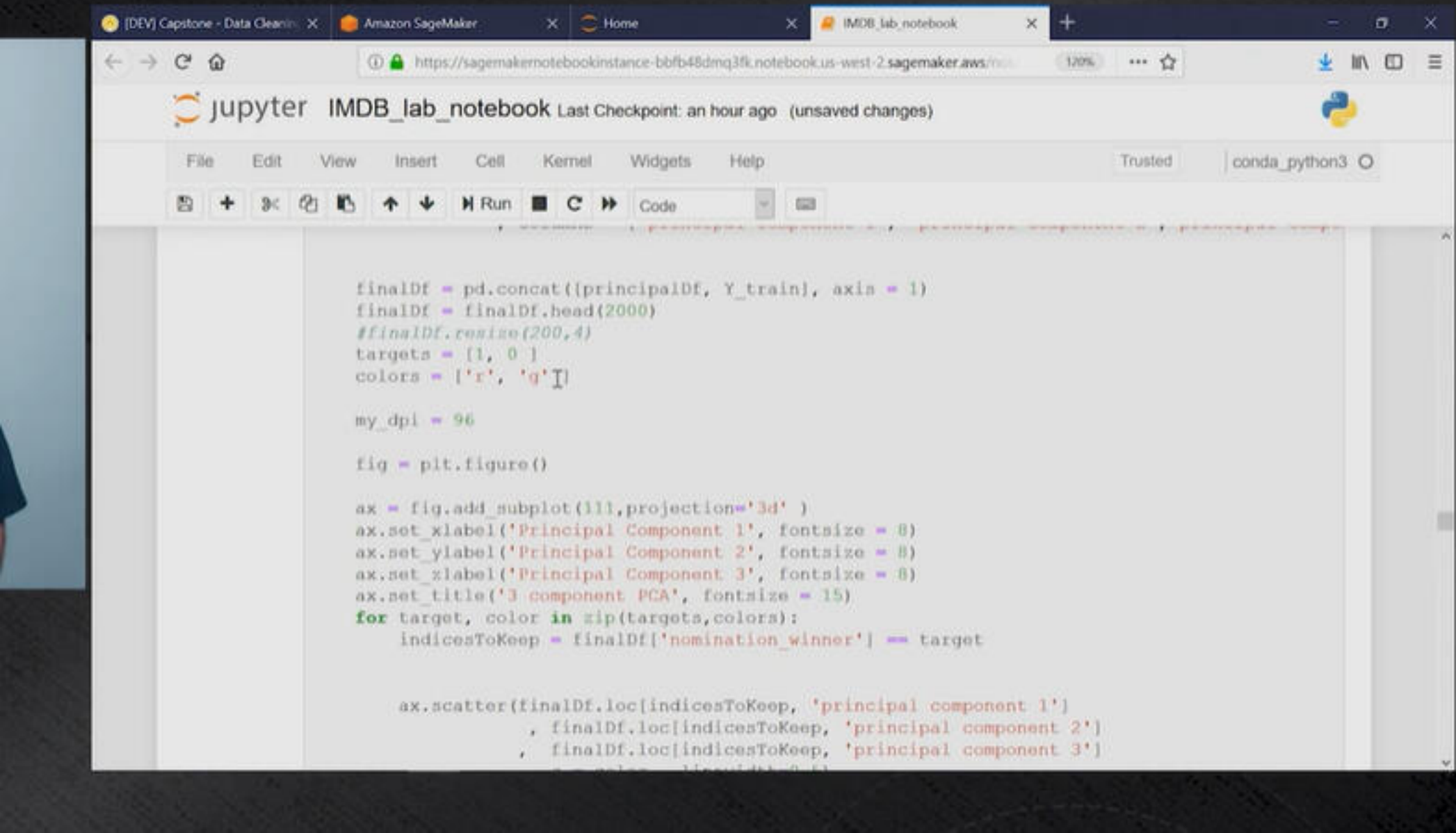
However, if all you're thinking about is petting furry animals and you try to pet the stray cat you see on the street, you may execute a plan that ruins your afternoon.

So while you engineer the features that matter to your business problem, remember to keep it simple

The information trail you follow depends on what you're paying attention to, and what you pay attention to *biases the patterns you construct.*

However, if all you're thinking about is petting furry animals and you try to pet the stray cat you see on the street, you may execute a plan that ruins your afternoon.

So while you engineer the features that matter to your business problem, remember to keep it simple and be cognizant of your attentional demands.


[DEV] Capstone - Data Clean
Amazon SageMaker
Home
IMDB_lab_notebook
https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Code
finalDf = pd.concat([principalDf, Y_train], axis = 1)
finalDf = finalDf.head(2000)
#finalDf.resize(200,4)
targets = [1, 0]
colors = ['r', 'g']
my_dpi = 96
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.set_xlabel('Principal Component 1', fontsize = 8)
ax.set_ylabel('Principal Component 2', fontsize = 8)
ax.set_zlabel('Principal Component 3', fontsize = 8)
ax.set_title('3 component PCA', fontsize = 15)
for target, color in zip(targets,colors):
indicesToKeep = finalDf['nomination_winner'] == target
ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
, finalDf.loc[indicesToKeep, 'principal component 2']
, finalDf.loc[indicesToKeep, 'principal component 3']

[DEV] Capstone - Data Clean | Amazon SageMaker | Home | IMDB_lab_notebook

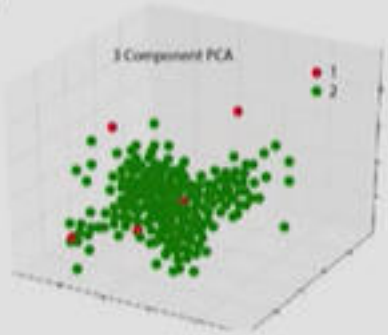
https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
ax.set_xlabel('Principal Component 1', fontsize = 8)
ax.set_ylabel('Principal Component 2', fontsize = 8)

ax.set_title('2 component PCA', fontsize = 15)
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['nomination_winner'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color,  linewidth=0.5)

ax.legend(targets)
#ax.grid()
plt.show()
plt.close()

X_train = pca.transform(X_train)
```



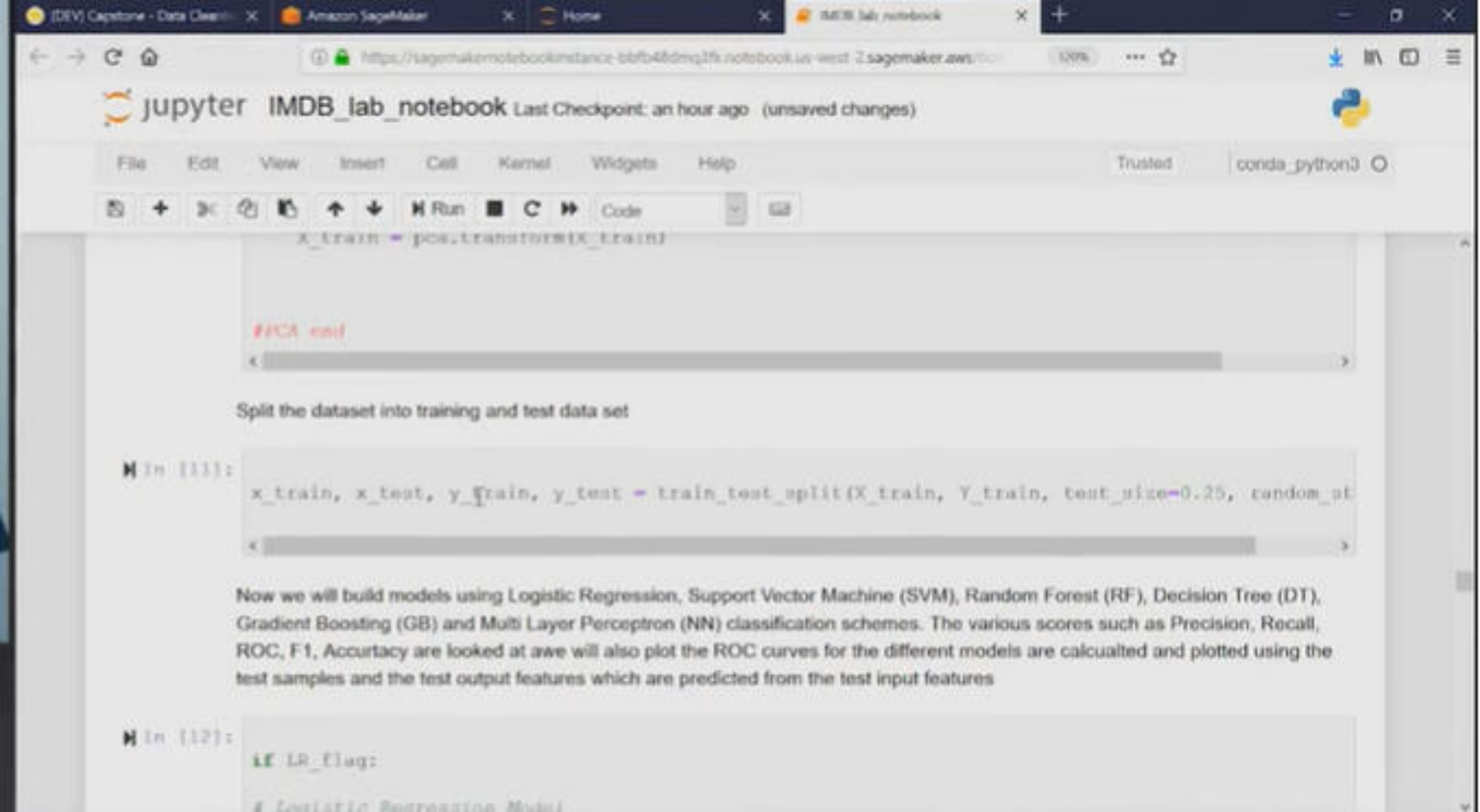

```
#PCA end
```

Split the dataset into training and test data set

Algorithm Comparison and Selection

As babies get older and their diversity of attention grows, they are able to optimize. They can determine: things I *actually* want to eat, things that are *actually* dangerous, animals I *actually* want to pet. Knowing what's best only comes with the experience of trying things over and over, comparing them, and then making a decision.

So now that you've cleaned your data, selected your features, and engineered some features, it's time to run a couple of models and then pick the best one for your business challenge.


jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Code
x_train = pca.transform(X_train)
#PCA end
Split the dataset into training and test data set
In [11]:
x_train, x_test, y_train, y_test = train_test_split(X_train, Y_train, test_size=0.25, random_st
Now we will build models using Logistic Regression, Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Gradient Boosting (GB) and Multi Layer Perceptron (NN) classification schemes. The various scores such as Precision, Recall, ROC, F1, Accurtacy are looked at awe will also plot the ROC curves for the different models are calcualted and plotted using the test samples and the test output features which are predicted from the test input features
In [12]:
if LR_flag:
# Logistic Regression Model

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

Gradient Boosting (GB) and Multi Layer Perceptron (NN) classification schemes. The various scores such as Precision, Recall, ROC, F1, Accurtacy are looked at awe will also plot the ROC curves for the different models are calcualted and plotted using the test samples and the test output features which are predicted from the test input features

In [12]:

```
if LR flag:

# Logistic Regression Model

    logisticRegr = LogisticRegression(random_state=0, solver='lbfgs',
                        multi_class='multinomial')
    logisticRegr.fit(x_train, y_train)
    y_test_pred_LR = cross_val_predict(logisticRegr, x_test, y_test, cv=3)
    score = logisticRegr.score(x_test, y_test)
    print("LR Accuracy", score)
    roc = roc_auc_score(y_test, y_test_pred_LR)
    print("roc score", roc)
    y_test_pred = y_test_pred_LR
    x = precision_score(y_test, y_test_pred)
    y = recall_score(y_test, y_test_pred)
    z = f1_score(y_test, y_test_pred)
    print("Precision ", x)
    print("recall ", y)
    print("f1 score ", z)
```

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
Y_train = df['nomination_winner']
# x_train, x_test, y_train, y_test = train_test_split(X_train, Y_train, test_size=0.25, random_

if normalize_flag:
    X_train=(StandardScaler().fit_transform(X_train ))
```

Experiment with Principal Component Aanlysis and reduce the Feature set to three main components: However there is no benfit in choosing PCA by looking at the roc scores, they infact go down. Also, looking at the 3D plot we see that that the "nomination/winner" 3D points (Red dots) are within the clusters of the non "nomination/winner" 3D points and seem to be suggesting that they are hard to be classified.

In [10]:

```
#PCA


if feature_pca_3D:
    pca = decomposition.PCA(n_components=3)



    principalComponents = pca.fit_transform(X_train )
```

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

"nomination/winner" 3D points (Red dots) are within the clusters of the non "nomination/winner" 3D points and seem to be suggesting that they are hard to be classified.

In [10]:
```
#PCA


if feature_pca_3D:
    pca = decomposition.PCA(n_components=3)




    principalComponents = pca.fit_transform(X_train )

    principalDf = pd.DataFrame(data = principalComponents
                 , columns = ['principal component 1', 'principal component 2','principal compo

    finalDf = pd.concat([principalDf, Y_train], axis = 1)
    finalDf = finalDf.head(2000)
    #finalDf.resize(200,4)
    targets = [1, 0 ]
    colors = ['r', 'g' ]
```

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
finalDf = pd.concat([principalDf, Y_train], axis = 1)
finalDf = finalDf.head(2000)
#finalDf.resize(200,4)
targets = [1, 0 ]
colors = ['r', 'g' ]

my_dpi = 96

fig = plt.figure()

ax = fig.add_subplot(111,projection='3d' )
ax.set_xlabel('Principal Component 1', fontsize = 8)
ax.set_ylabel('Principal Component 2', fontsize = 8)
ax.set_zlabel('Principal Component 3', fontsize = 8)
ax.set_title('3 component PCA', fontsize = 15)
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['nomination_winner'] == target


    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , finalDf.loc[indicesToKeep, 'principal component 3']
```

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
print("roc socre", roc)
y_test_pred = y_test_pred_LR
x = precision_score(y_test, y_test_pred)
y = recall_score(y_test, y_test_pred)
z = f1_score(y_test, y_test_pred)
print("Precision ", x)
print("recall ", y)
print("f1 score ", z)
# Plot   confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

plot_confusion_matrix(cnf_matrix, classes=['Class 0', 'Class 1'],
                      title='Confusion matrix, without normalization')

# Compute ROC curve and ROC area
#fpr, tpr, threshold = roc_curve(y_test , y_test_pred)



logit_roc_auc = roc_auc_score(y_test_pred, logisticRegr.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test, logisticRegr.predict_proba(x_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression ' )
```

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Markdown

```
import pickle
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import gzip
#import statsmodels.formula.api as smf
import numpy as np
import seaborn as sns

sns.set()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import decomposition
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
```

https://sagemakernotebookinstance.bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
                        multi_class='multinomial')
logisticRegr.fit(x_train, y_train)
y_test_pred_LR = cross_val_predict(logisticRegr, x_test, y_test, cv=3)
score = logisticRegr.score(x_test, y_test)
print("LR Accuracy", score)
roc = roc_auc_score(y_test, y_test_pred_LR)
print("roc score", roc)
y_test_pred = y_test_pred_LR
x = precision_score(y_test, y_test_pred)
y = recall_score(y_test, y_test_pred)
z = f1_score(y_test, y_test_pred)
print("Precision ", x)
print("recall ", y)
print("f1 score ", z)
# Plot   confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

plot_confusion_matrix(cnf_matrix, classes=['Class 0', 'Class 1'],
                      title='Confusion matrix, without normalization')

# Compute ROC curve and ROC area
#fpr, tpr, threshold = roc_curve(y_test , y_test_pred)
```

https://sagemakernotebookinstance.bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
                        multi_class='multinomial')
logisticRegr.fit(x_train, y_train)
y_test_pred_LR = cross_val_predict(logisticRegr, x_test, y_test, cv=3)
score = logisticRegr.score(x_test, y_test)
print("LR Accuracy", score)
roc = roc_auc_score(y_test, y_test_pred_LR)
print("roc score", roc)
y_test_pred = y_test_pred_LR
x = precision_score(y_test, y_test_pred)
y = recall_score(y_test, y_test_pred)
z = f1_score(y_test, y_test_pred)
print("Precision ", x)
print("recall ", y)
print("f1 score ", z)
# Plot   confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

plot_confusion_matrix(cnf_matrix, classes=['Class 0', 'Class 1'],
                      title='Confusion matrix, without normalization')

# Compute ROC curve and ROC area
#fpr, tpr, threshold = roc_curve(y_test , y_test_pred)
```

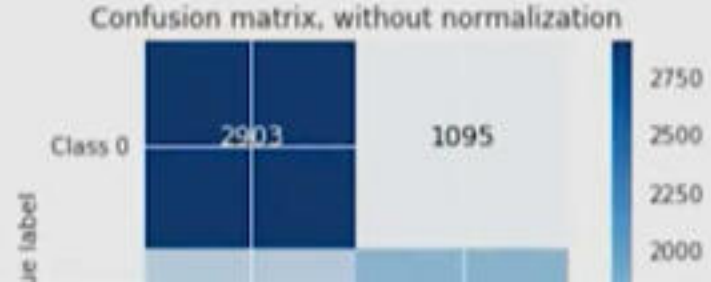
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

    plt.show()
```

```
LR Accuracy 0.6362522129919651
roc socre 0.6336095925391694
Precision  0.6230636833046471
recall  0.5411061285500748
f1 score  0.5791999999999999
Confusion matrix, without normalization
[[2903 1095]
 [1535 1810]]
```



https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

False Positive Rate

In [13]:

```
if SVM_flag:
    # SVM Model

    sgd_clf = SGDClassifier(random_state=0, loss="log" )
    sgd_clf.fit(x_train, y_train)
    cross_val_score(sgd_clf, x_train, y_train, cv=3, scoring="accuracy")
    y_test_pred_SVC = cross_val_predict(sgd_clf, x_test, y_test, cv=3)
    predictions = sgd_clf.predict(x_test)
    score = sgd_clf.score(x_test, y_test)
    print("SVM Accuracy", score)
    roc = roc_auc_score(y_test, y_test_pred_SVC)
    print("roc socre", roc)
    y_test_pred = y_test_pred_SVC
    x = precision_score(y_test, y_test_pred)
    y = recall_score(y_test, y_test_pred)
    z = f1_score(y_test, y_test_pred)
    print("Precision ", x)
    print("recall ", y)
    print("f1 score ", z)
    # Plot   confusion matrix
```

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
x = precision_score(y_test, y_test_pred)
y = recall_score(y_test, y_test_pred)
z = f1_score(y_test, y_test_pred)
print("Precision ", x)
print("recall ", y)
print("f1 score ", z)
# Plot   confusion matrix
cnf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

plot_confusion_matrix(cnf_matrix, classes=['Class 0', 'Class 1'],
                      title='Confusion matrix, without normalization')

# Compute micro-average ROC curve and ROC area
fpr, tpr, threshold = roc_curve(y_test , y_test_pred)



roc_auc = roc_auc_score(y_test_pred, sgd_clf.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test, sgd_clf.predict_proba(x_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='SVM  ' )
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
```

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
plt.figure()
plt.plot(fpr, tpr, label='SVM  ' )
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")

plt.show()
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/linear_model/stochastic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 'sklearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
```

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted conda_python3

Run Code

```
, and default tol will be 1e-3.
  type(self), FutureWarning)
/lib/python3.6/site-packages/sklearn/linear_model/stoch
max_iter and tol parameters have been added in <class 's
gradient.SGDClassifier'> in 0.19. If both are left unset, they
default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. F
rom 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/linear_model/stocha
stic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 's
klearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they
default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. F
rom 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/linear_model/stocha
stic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 's
klearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they
default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. F
rom 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/linear_model/stocha
stic_gradient.py:128: FutureWarning: max_iter and tol parameters have been added in <class 's
klearn.linear_model.stochastic_gradient.SGDClassifier'> in 0.19. If both are left unset, they
default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. F
```

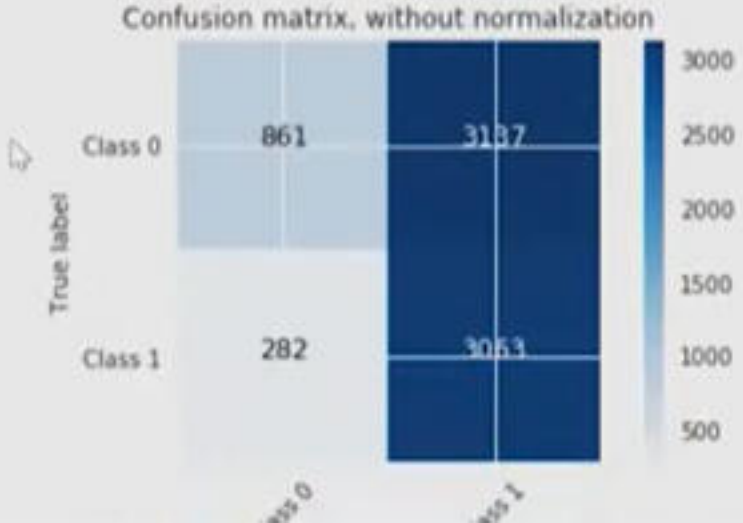
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
SVM Accuracy 0.6200463026011167
roc socre 0.5655263730519968
Precision  0.4940322580645161
recall  0.915695067264574
f1 score  0.6418019905709795
Confusion matrix, without normalization
[[ 861 3137]
 [ 282 3063]]
```



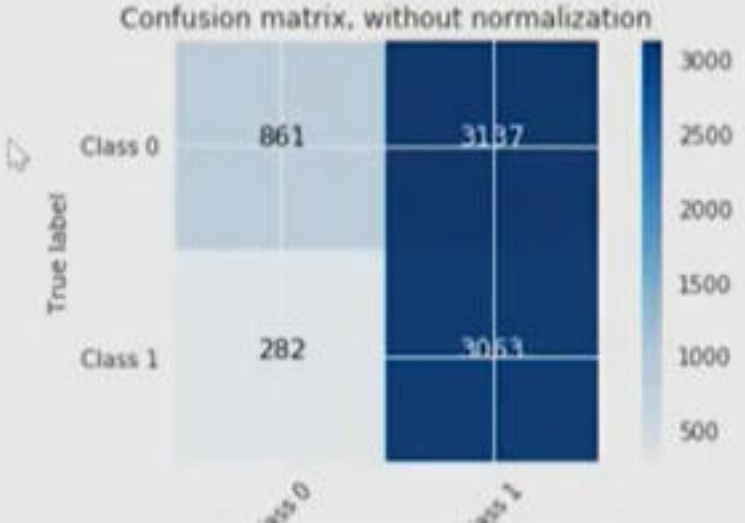
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

```
SVM Accuracy 0.6200463026011167
roc socre 0.5655263730519968
Precision  0.4940322580645161
recall  0.915695067264574
f1 score  0.6418019905709795
Confusion matrix, without normalization
[[ 861 3137]
 [ 282 3063]]
```



[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

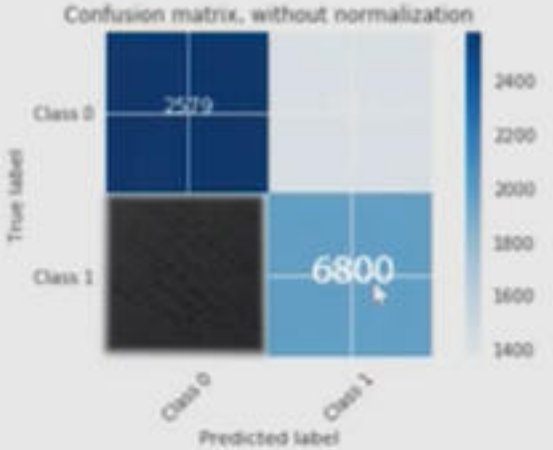
https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
roc socre 0.6174540558769668
Precision  0.5816627358490566
recall  0.5898355754857997
f1 score  0.5857206471723319
Confusion matrix, without normalization
[[2579 1419]
 [1372 1973]]
```



[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

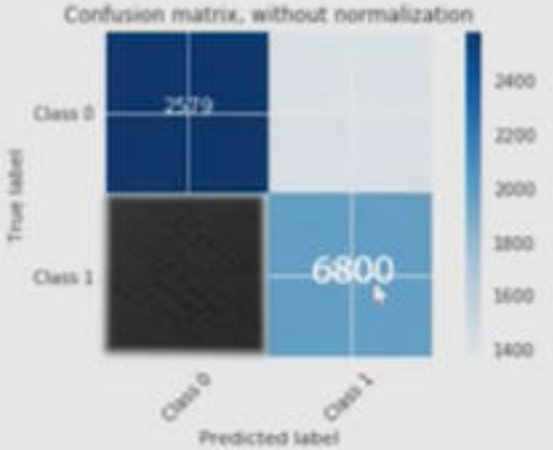
https://sagemakernotebookinstance-bbfb48dmq1fk.notebook.us-west-2.sagemaker.aws/

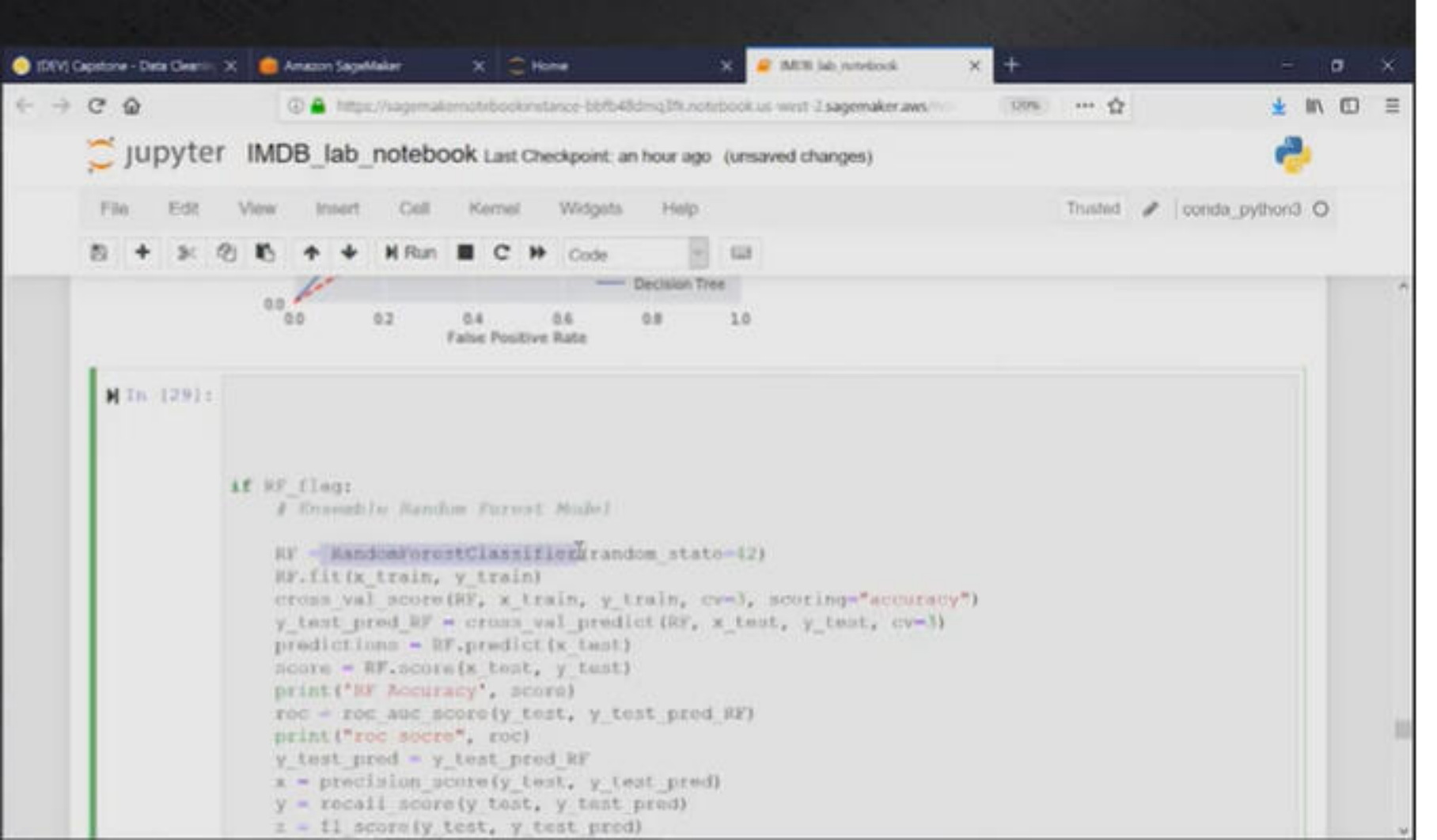
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
roc socre 0.6174540558769668
Precision  0.5816627358490566
recall  0.5898355754857997
f1 score  0.585720647172331 9
Confusion matrix, without normalization
[[2579 1419]
 [1372 1973]]
```




jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted conda_python3
Run Code
Decision Tree
0.0
0.0 0.2 0.4 0.6 0.8 1.0
False Positive Rate
In [29]:
if RF_flag:
# Ensemble Random Forest Model
RF = RandomForestClassifier(random_state=42)
RF.fit(x_train, y_train)
cross_val_score(RF, x_train, y_train, cv=3, scoring="accuracy")
y_test_pred_RF = cross_val_predict(RF, x_test, y_test, cv=3)
predictions = RF.predict(x_test)
score = RF.score(x_test, y_test)
print('RF Accuracy', score)
roc = roc_auc_score(y_test, y_test_pred_RF)
print("roc score", roc)
y_test_pred = y_test_pred_RF
x = precision_score(y_test, y_test_pred)
y = recall_score(y_test, y_test_pred)
z = f1_score(y_test, y_test_pred)

[DEV] Capstone - Data Clean | Amazon SageMaker | Home | IMDB_lab_notebook

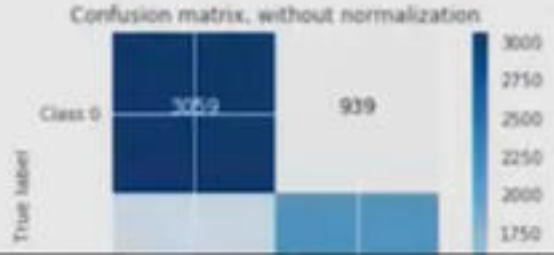
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

    plt.show()
```

```
GB Accuracy 0.7084297970856598
roc socre 0.7016993175212 42
Precision  0.6945348080676643
recall  0.6302660687593423
f1 score  0.6652126499454744
Confusion matrix, without normalization
[[3059  939]
 [1210 2135]]
```



[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

https://sagemakernotebookinstance-bbfb48dmq3fi.notebook.us-west-2.sagemaker.aws/

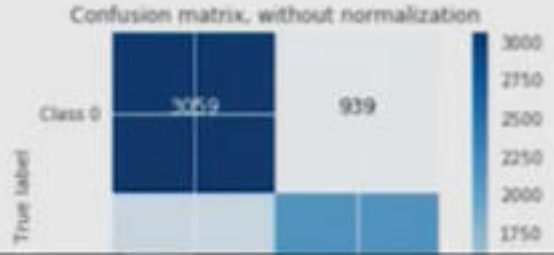
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

    plt.show()
```

```
GB Accuracy 0.7[illegible]297970856598
roc socre 0.701699317521242
Precision  0.6945348080676643
recall  0.6302660687593423
f1 score  0.6652126499454744
Confusion matrix, without normalization
[[3059  939]
 [1210 2135]]
```



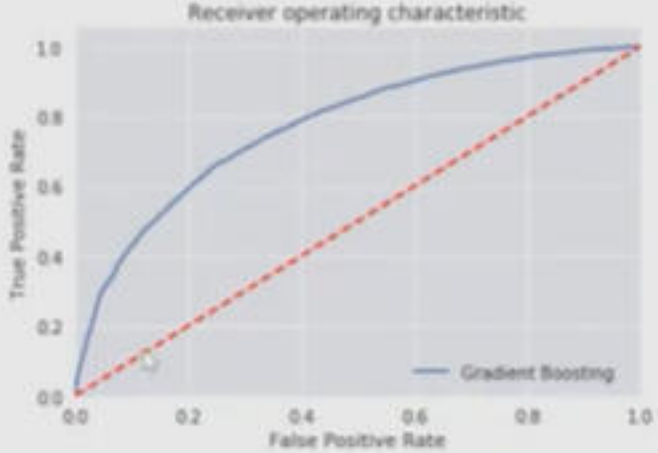
[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

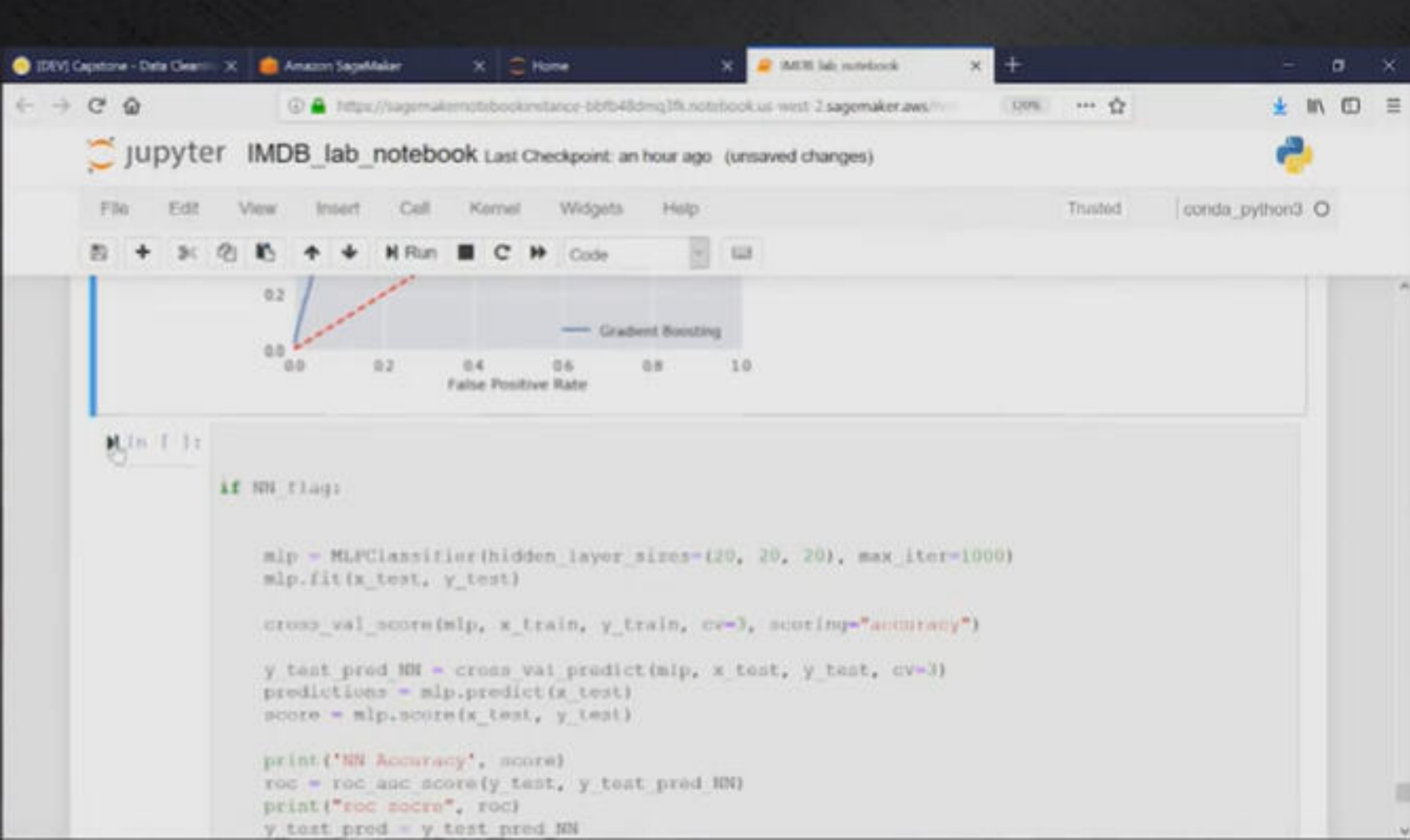
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code



In [ ]:


jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted conda_python3
Run Code
0.2
0.0
Gradient Boosting
0.0 0.2 0.4 0.6 0.8 1.0
False Positive Rate
In [ ]:
if NN_flag:
mlp = MLPClassifier(hidden_layer_sizes=(20, 20, 20), max_iter=1000)
mlp.fit(x_test, y_test)
cross_val_score(mlp, x_train, y_train, cv=3, scoring="accuracy")
y_test_pred_NN = cross_val_predict(mlp, x_test, y_test, cv=3)
predictions = mlp.predict(x_test)
score = mlp.score(x_test, y_test)
print('NN Accuracy', score)
roc = roc_auc_score(y_test, y_test_pred_NN)
print("roc socre", roc)
y_test_pred = y_test_pred_NN

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
print("Precision ", x)
print("Recall ", y)
print("f1 score ", z)
# plot confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred)
np.set_printoptions(precision=2)

plot_confusion_matrix(cnf_matrix, classes=['Class 0', 'Class 1'],
                      title='Confusion matrix, without normalization')

# Compute micro-average ROC curve and ROC area
fpr, tpr, threshold = roc_curve(y_test , y_test_pred)
roc_auc = auc(fpr , tpr )



# auc = roc_auc_score(y_test_pred, mlp.predict(x_test))
fpr, tpr, thresholds = roc_curve(y_test, mlp.predict_proba(x_test)[:, 1])
plt.figure()
plt.plot(fpr, tpr, label='MLP' % )
```

[DEV] Capstone - Data Clean | Amazon SageMaker | Home | IMDB lab notebook

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
```

```
NN Accuracy 0.5817705646193654
roc score 0.5800341501094335
Precision  0.5864391558149977
recall  0.39043368701016443
f1 score  0.4687724335965542
Confusion matrix, without normalization
[[3077  921]
 [2039 1306]]
```

[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

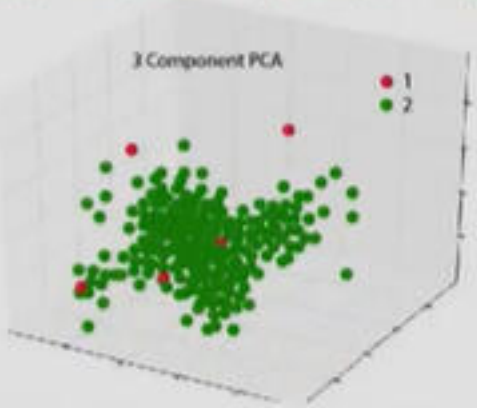
https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

```
if feature_pca_3D:
    pca = decomposition.PCA(n_components=3)



    principalComponents = pca.fit_transform(X_train )

    principalDf = pd.DataFrame(data = principalComponents
                 , columns = ['principal component 1', 'principal component 2','principal compo


    finalDf = pd.concat([principalDf, Y_train], axis = 1)
    finalDf = finalDf.head(2000)
    #finalDf.resize(200,4)
    targets = [1, 0 ]
    colors = ['r', 'g' ]

    my_dpi = 96

    fig = plt.figure()

    ax = fig.add_subplot(111,projection='3d' )
    ax.set_xlabel('Principal Component 1', fontsize = 8)
    ax.set_ylabel('Principal Component 2', fontsize = 8)
```



[DEV] Capstone - Data Clean... | Amazon SageMaker | Home | IMDB_lab_notebook

https://sagemakernotebookinstance-bbfb48dmq3fk.notebook.us-west-2.sagemaker.aws/

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

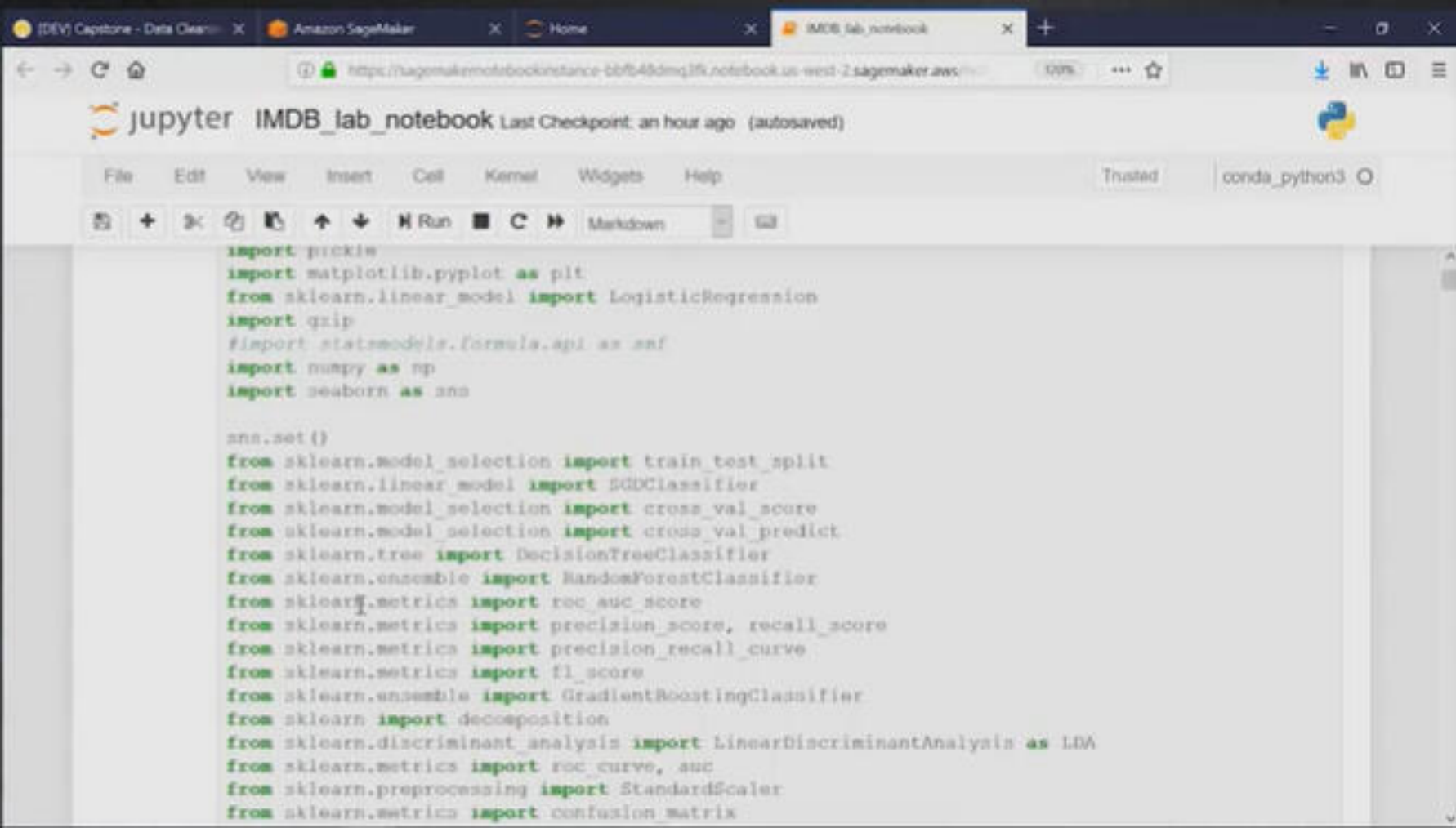
Run Code

experiment with different features and look at the roc numbers

In [23]:

```
# features to retain
if feature_premiere_wide:
    X_train = df[['rating', 'ratingCount', 'runtimeMinutes','premiere','wide']]
elif feature_premiere:
    X_train = df[['rating', 'ratingCount', 'runtimeMinutes','premiere' ]]
elif feature_wide:
    X_train = df[['rating', 'ratingCount',  'runtimeMinutes' ,'wide']]
else :
    X_train = df[['rating', 'ratingCount',  'runtimeMinutes']]


# X_train = df[  'ratingCount',  'runtimeMinutes', 'premiere' ]]
Y_train = df['nomination_winner']
# x_train, x_test, y_train, y_test = train_test_split(X_train, Y_train, test_size=0.25, random_

if normalize_flag:
    X_train=(StandardScaler().fit_transform(X_train ))
```

Experiment with Principal Component Aanlysis and reduce the Feature set to three main components: However there is no benfit


[DEV] Capstone - Data Clean
Amazon SageMaker
Home
IMDB_lab_notebook
jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Markdown
import pickle
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import gzip
#import statsmodels.formula.api as smf
import numpy as np
import seaborn as sns
sns.set()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import decomposition
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix

jupyter IMDB_lab_notebook Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code

In [*]:

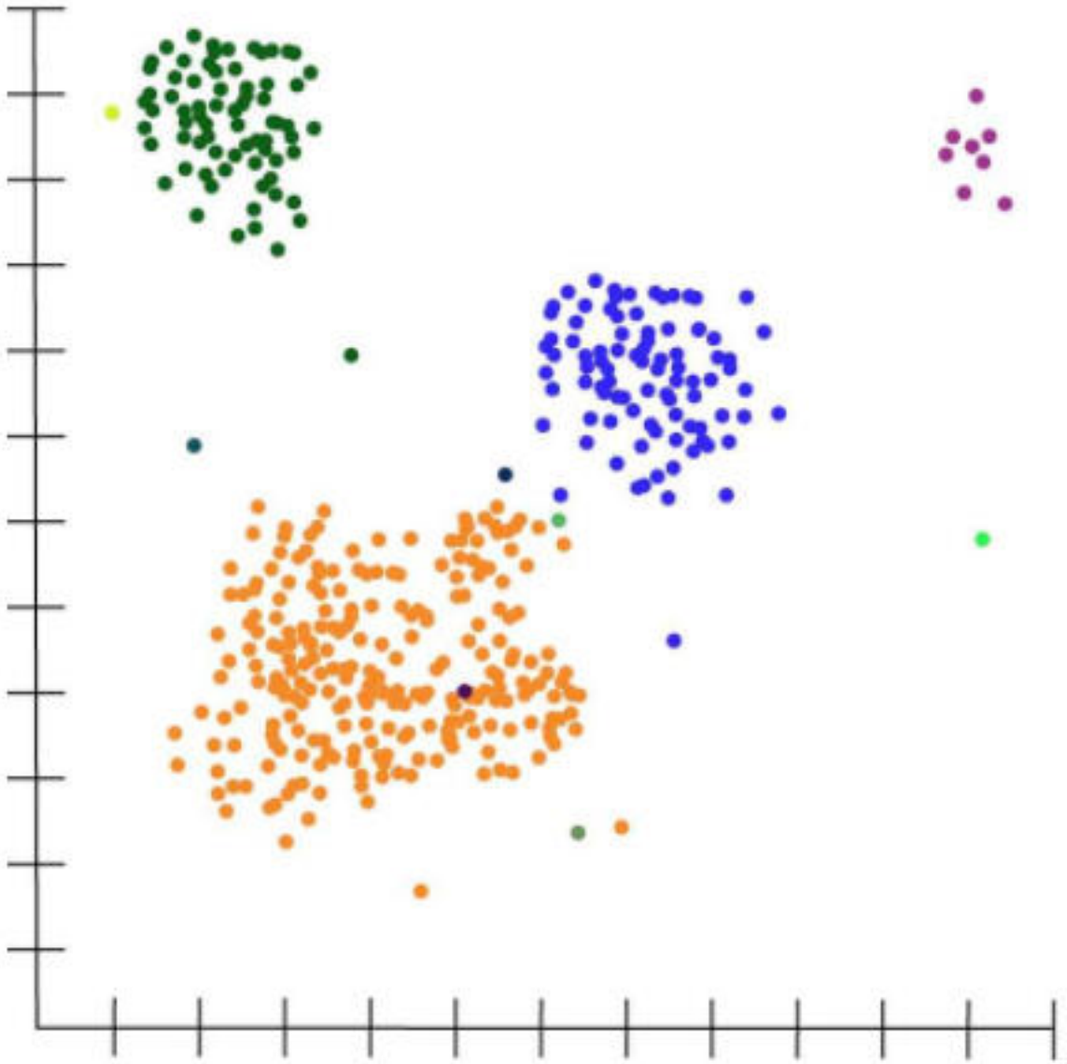
```
if NN_flag:


    mlp = MLPClassifier(hidden_layer_sizes=(20, 20, 20), max_iter=1000)
    mlp.fit(x_test, y_test)

    cross_val_score(mlp, x_train, y_train, cv=3, scoring="accuracy")

    y_test_pred_NN = cross_val_predict(mlp, x_test, y_test, cv=3)
    predictions = mlp.predict(x_test)
    score = mlp.score(x_test, y_test)

    print('NN Accuracy', score)
    roc = roc_auc_score(y_test, y_test_pred_NN)
    print("roc socre", roc)
    y_test_pred = y_test_pred_NN
```

Defining Your Business Problem

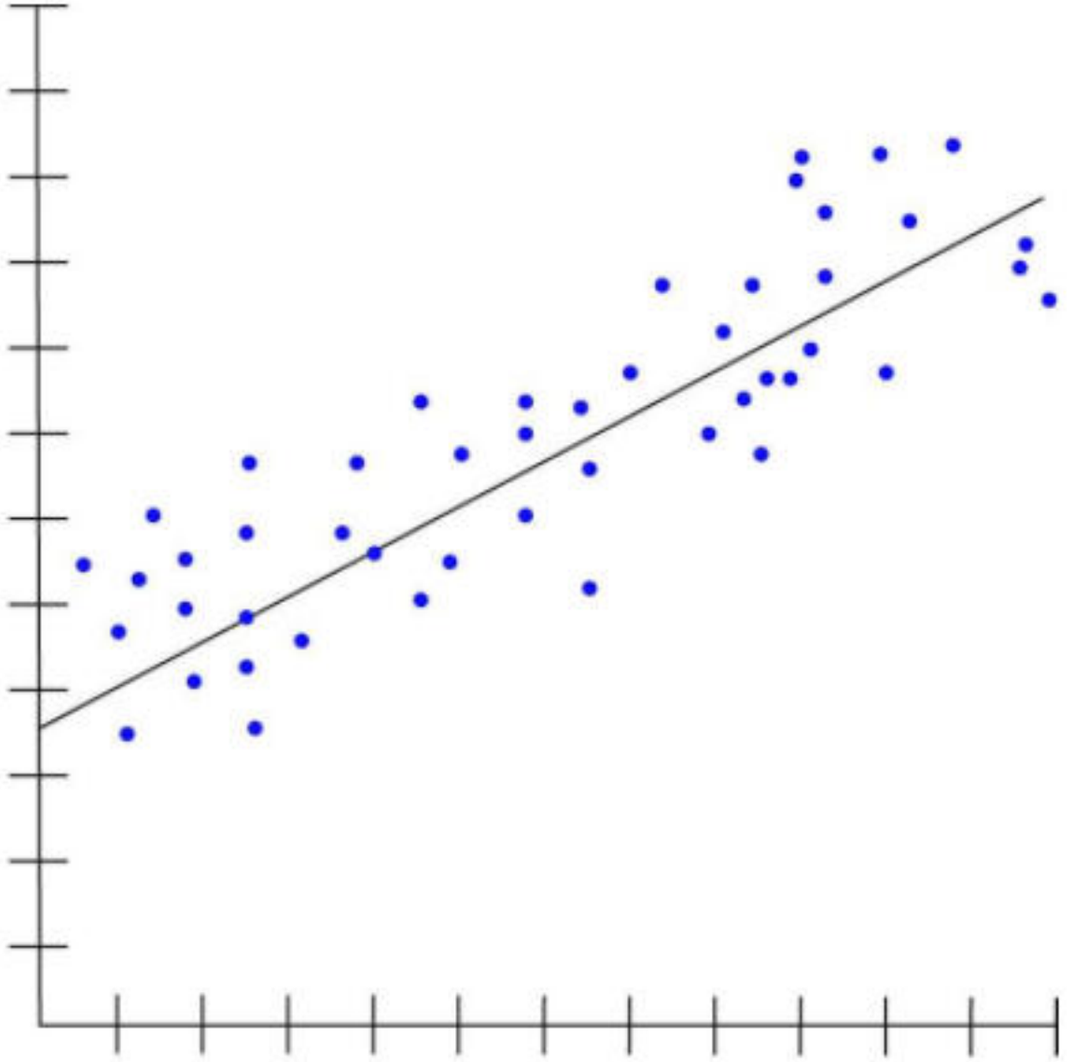
The first step to applying a machine learning solution is having a clearly defined business problem. Understanding what you're solving for drives how you solve for it. What data makes sense for one model will likely not make perfect sense for another, because the right tool for the job depends on knowing what the job actually is. For instance, wanting to know how to sell more of your product is not the same as wanting to know how much of your product you need in stock to be prepared for the upcoming holiday.

In the first example, you want to market directly to people with the highest likelihood to buy, which means you need to rank or cluster your customers from lowest to highest buying potential. Data that could help you understand how to rank your customers may include past purchasing data, ad clicks, or traffic data. Contrast this clustering problem with the second example where you want to predict purchases for a specific period of time in the future. Data that could help you with this regression problem may be historical data, data on similar products, or store information.

In the first example, you want to market directly to people with the highest likelihood to buy, which means you need to rank or cluster your customers from lowest to highest buying potential. Data that could help you understand how to rank your customers may include past purchasing data, ad clicks, or traffic data. Contrast this clustering problem with the second example where you want to predict purchases for a specific period of time in the future. Data that could help you with this regression problem may be historical data, data on similar products, or store information.



How do we sell more product?



How much product will we sell?

Problem Definition

Amazon Studios is known for their bold and innovative original productions. They produce series and films from top tier and up-and-coming creators for customers in over 200 countries and territories. Let's hear about how a team like theirs could use machine learning to help solve one of their business problems.

Suppose You Work for Amazon Studios

YOUR ROLE | THE DATA | YOUR TASK

Assuming the role of lead data scientist in 2005, you're presented with a challenge: Amazon Studios wants to produce award-winning films and, therefore, focus the budget on projects with the best chance of winning those awards. Using the ***actual dataset*** from IMDb, an Amazon subsidiary, for movies made between 1990 and 2005, you begin your investigation.

amazon studios

YOUR ROLE | THE DATA | YOUR TASK

The IMDb dataset is a feature-rich, comprehensive listing of all films released during that time period; it includes critical data such as cast and crew, synopsis, and other production data. Much of this data is published on the public IMDb.com site, while other features are embargoed for studio analytics.

amazon studios

YOUR ROLE	THE DATA	YOUR TASK

Your task is to predict which movies will most likely be nominated for an award during the "upcoming" 2005 awards season by building an awards analysis prediction model. To do that, you can turn your attention to the question of "what data would make sense for my model?"

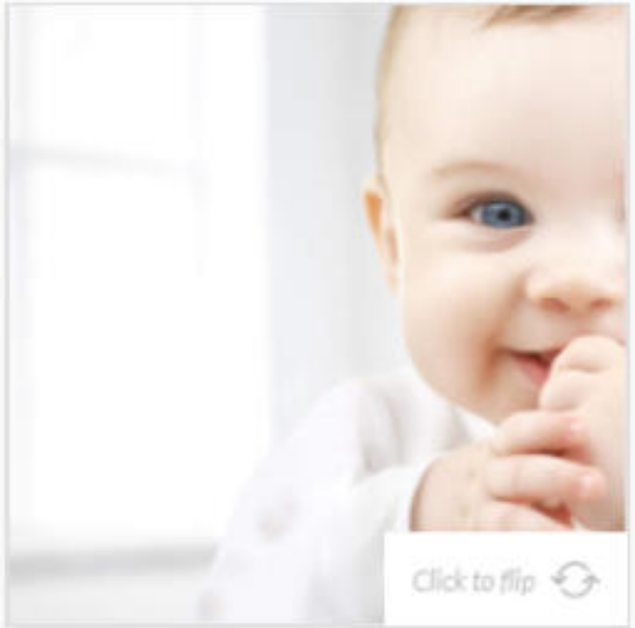
Thinking About Your Data

The question of "what data makes sense for my model?" is a familiar one to many teams inside Amazon. Responsible for developing the technology behind Alexa, Amazon Go, and Amazon.com's recommendation system, Amazonians are pushing the bounds of machine learning innovation. When it comes to datasets there are several things to consider, many of which are of equal importance. How many data points do you need – thousands? Millions? Is your formatting the same for all dates – 05/22/18 or 22/05/18? Are dollar amounts using the same currency? Do you need labeled data? What do you do with missing data? Is the data possibly biased?

To aggregate the various elements of managing data, consider the following analogy.

03/22/18 or 22/03/18? Are dollar amounts using the same currency? Do you need labeled data? What do you do with missing data? Is the data possibly biased?

To aggregate the various elements of managing data, consider the following analogy.



It is through this neural engine that features are kept or discarded, deemed useful or not.

It is through this neural engine that features are kept or discarded, deemed useful or not.

This is no small task. Half of a human's visual brain power is responsible for processing less than 5% of our visual world. It's impossible for our optical nerves to take in all the data the world has to offer at once – give it a try! Without moving your eyes, how many data points can you observe? Now try again, but this time move your eyes 5 times turning your head if you need to. This is how we are able to focus our brain power.

Where we look and what we pull from patterns, dictates what we see.

---

patterns, dictates what we see.

Finding patterns among features is how we make sense of the world but when we are born, we aren't programmed to automatically recognize cars or cities. As babies we take visual information and apply it to non-visual concepts. We learn to find the path in front of us by bumping into things; we learn about emotion by watching faces and integrating facial expressions with spoken words. These patterns are systematically built upon our attentional demands and repeated exposure. The same can be said for machine learning whereby focusing on the data that you believe makes the most sense up front, running tests on that data, adjusting it, running more tests, and adjusting it again, your model will build this same awareness.

The Notebook, and Cleaning and Visualizing Your Data

Through the eyes of a child the world is a conglomerate of textures, gradients, heights, shadows, contrasts, and depths. A baby might stumble across a copy the Wall Street Journal and discard any attempt to parse the written content covering international finance regulations because that is irrelevant to her world model. However, she will catalog the features of the newspaper as “big”, “fun to grab”, and “not tasty”.

Sifting through all of this data to find patterns and associate meaning takes time, *but to find a pattern is to find a solution.*

To do this successfully, both as a child and in machine learning, you need to collect as much data as possible and then reduce the amount of fragmented data. You form a hypothesis and say to yourself, "I think I'll need these features" so you gather that data, conduct your experiment, get your results, and adjust your features accordingly – deciding what data helps your model and what is a distractor. And you do this repeatedly. With regard to machine learning, these processes are formally referred to as cleaning and visualizing your data.

To do this successfully, both as a child and in machine learning, you need to collect as much data as possible and then reduce the amount of fragmented data. You form a hypothesis and say to yourself, "I think I'll need these features" so you gather that data, conduct your experiment, get your results, and adjust your features accordingly – deciding what data helps your model and what is a distractor. And you do this repeatedly. With regard to machine learning, these processes are formally referred to as cleaning and visualizing your data.

Let's get started with an introduction to the notebook we'll be using, and then dive right into the processes of cleaning and visualizing the data.

Algorithm Comparison and Selection

As babies get older and their diversity of attention grows, they are able to optimize. They can determine: things I *actually* want to eat, things that are *actually* dangerous, animals I *actually* want to pet. Knowing what's best only comes with the experience of trying things over and over, comparing them, and then making a decision.

So now that you've cleaned your data, selected your features, and engineered some features, it's time to run a couple of models and then pick the best one for your business challenge.