Project Presentation

Zhe Wu
(zw1629)

Yiqiu Wu
(yw3101)

Background

- Objective:
  - predict the Up or Down of the currency pair EUR/USD
- Data:
  - Historical data with the information from 2008-01-01 to 2018-03-19
- Method:
  - Logistic regression
  - SVM
  - Decision tree
  - Random forest
  - Gradient-Boosted Tree
  - Multilayer perceptron classifier

Description of the Dataset

- Forex Market Dataset
  - 5-min Bid price of EUR/USD
- Time range:
  - 2008-01-01 to 2018-03-19
- Training data size: 49652
- Test data size: 10,962
- Labels
  - 1 class (0 or 1), representing Up or Down of this currency pair
- Number of features: 212

Methodology - Logistic regression

```
colname = data.columns
colname.remove('_c0')
colname.remove('up_down')
feature = VectorAssembler(inputCols = colname,outputCol="features")
data2 = feature.transform(data)
normalizer = Normalizer(inputCol="features", outputCol="normFeatures", p=1.0)
normdata = normalizer.transform(data2)
output = normdata.selectExpr("up_down as label", "normFeatures as features")
```

In [7]:

```
train, test = output.randomSplit([0.7, 0.3], seed = 300)
lr = LogisticRegression(featuresCol = "features", labelCol = "label", maxIter=100)
lrModel = lr.fit(train)
predictions = lrModel.transform(test)
```

Methodology - SVM

- An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

NYU

Methodology - SVM

- Coding for SVM is done using scikit-learn package of Python.

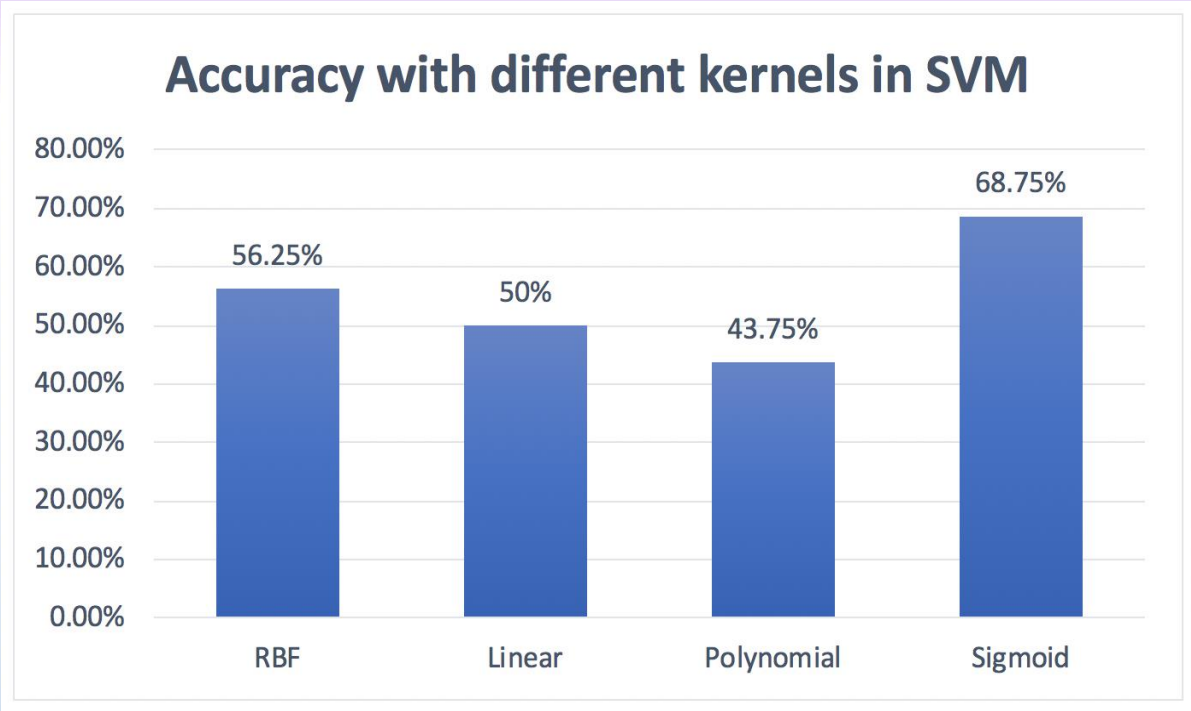
```
def getData():
    data, label = readFile()
    sss = StratifiedShuffleSplit(n_splits=5, test_size=0.5, random_state=0)
    sss.get_n_splits(data, label)
    for train_index, test_index in sss.split(data, label):
        X_train, X_test = data[train_index], data[test_index]
        y_train, y_test = label[train_index], label[test_index]
    return X_train, X_test, y_train, y_test
```

NYU

Methodology - SVM

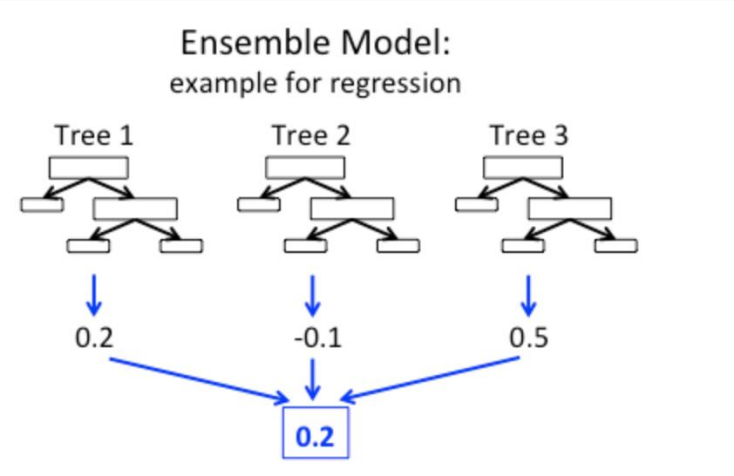
```
def compare(list1,list2):
    correctTimes = 0.0
    for i in range(0,len(list1)):
        if list1[i] == list2[i]:
            correctTimes += 1
    accuracy = correctTimes/len(list1)
    return accuracy

def getAccuracy(X_train, X_test, y_train, y_test):
    rbf_svc = svm.SVC(kernel='sigmoid',C = 10, gamma= 0.1)
    rbf_svc.fit(X_train,y_train)
    prediction = rbf_svc.predict(X_test)
    return compare(prediction,y_test)

# 5-fold Cross Validation
avgAccuracy = []
for count in range(1,6):
    print ('----------------------------count = '+str(count) + '----------------------------')
    X_train, X_test, y_train, y_test = getData()
    avgAccuracy.append(getAccuracy(X_train, X_test, y_train, y_test))
```

Methodology - SVM



Decision Tree, Random Forests & Gradient-Boosted Trees

Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions. Tree ensemble algorithms such as random forests and boosting are among the top performers for classification and regression tasks.



Methodology - Gradient-boosted Tree

Coding for Gradient-boosted tree is done using Pyspark

```
input_data = VectorAssembler(inputCols= colname ,outputCol="features")
output_data = input_data.transform(GBT)
```

```
labelIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel").fit(output_data3)
featureIndexer = VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(output_data3)

# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = output_data3.randomSplit([0.7, 0.3])

# Train a GBT model.
gbt = GBTClassifier(labelCol="indexedLabel", featuresCol="indexedFeatures", maxIter=100)
```

NYU

Methodology - Gradient-boosted Tree

```
# Chain indexers and GBT in a Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, gbt])

# Train model.  This also runs the indexers.
model = pipeline.fit(trainingData)

# Make predictions.
predictions = model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "indexedLabel", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(
    labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
```

Methodology – Multilayer Perceptron Classifier

```
# Split the data into train and test
nn_colname = nn_data.columns
nn_colname.remove('_c0')
nn_colname.remove('up_down')
nn_feature = VectorAssembler(inputCols = nn_colname,outputCol="features1")
nn_data2 = feature.transform(rf_data)
nn_normalizer = Normalizer(inputCol="features1", outputCol="normFeatures", p=1.0)
nn_normdata = nn_normalizer.transform(nn_data2)
nn_output = nn_normdata.selectExpr("up_down as label", "normFeatures as features")
train, test = nn_output.randomSplit([0.6, 0.4], 1234)
```

In [22]:

```
# specify layers for the neural network:
# input layer of size 4 (features), two intermediate of size 5 and 4
# and output of size 3 (classes)
layers = [212, 100, 50, 10, 2]
```

In [23]:

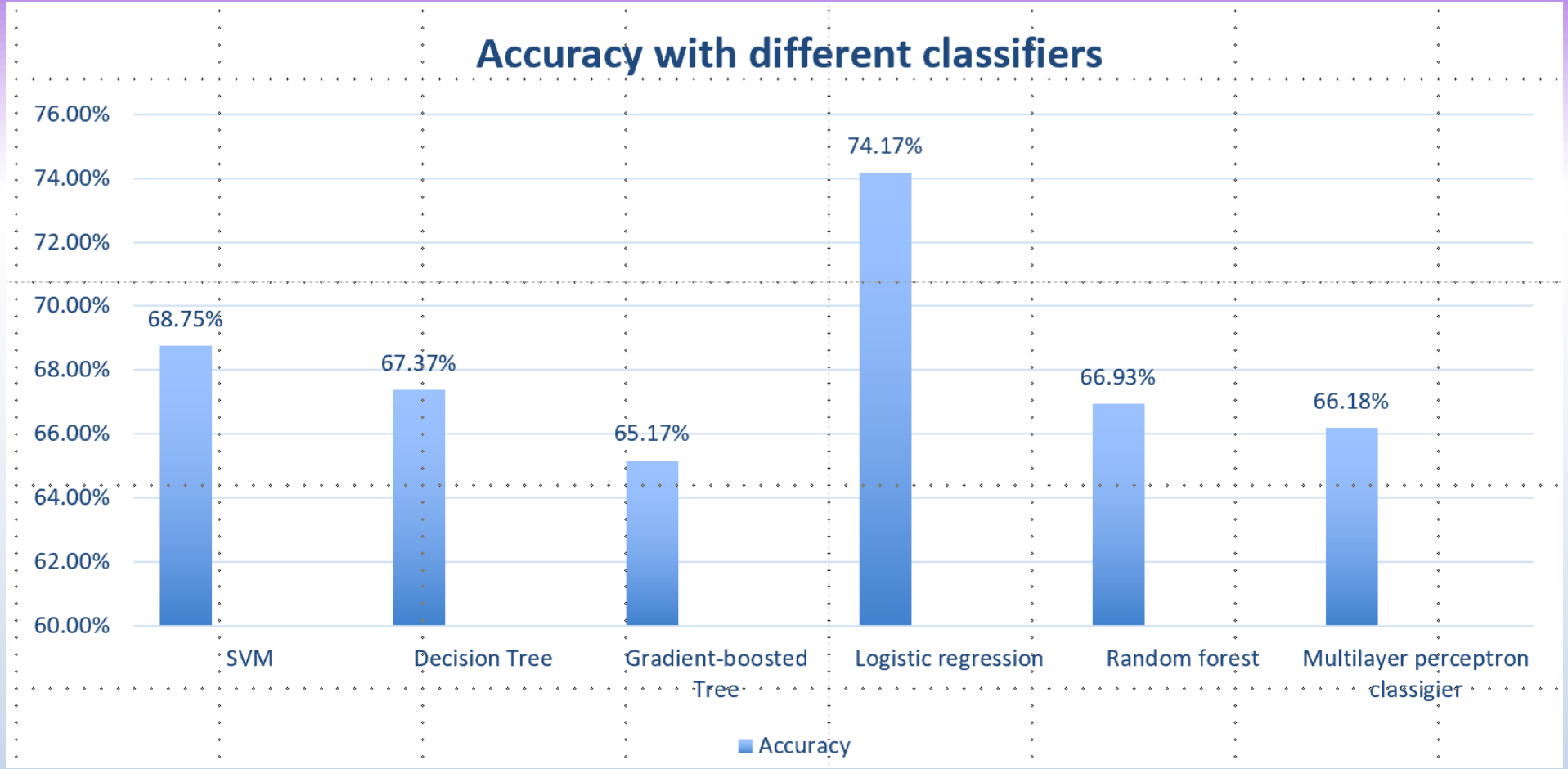
```
# create the trainer and set its parameters
trainer = MultilayerPerceptronClassifier(maxIter=100, layers=layers, blockSize=128, seed=1234)
```

In [25]:

```
# train the model
model = trainer.fit(train)
```

NYU

Result


Accuracy with different classifiers
76.00%
74.00%
72.00%
70.00%
68.00%
66.00%
64.00%
62.00%
60.00%
68.75%
67.37%
65.17%
74.17%
66.93%
66.18%
SVM
Decision Tree
Gradient-boosted Tree
Logistic regression
Random forest
Multilayer perceptron classigier
Accuracy

Conclusion

- The logistic regression method has the highest accuracy among these different classifiers. Therefore, we may adopt this method to predict changes in exchange rates.
- The random forest and multilayer perception classifier are the methods that we never touched before. However, asked one friend who has professional knowledges in financial engineering area, we noticed that they are also common and effective approach to predict exchange rate. So we compare with them as well.
- We are relatively familiar with logistic regression and support vector machine. By doing this project, we successfully applied the knowledge learned in the classroom and were able to process the data correctly. We learned a lot from machine learning.

Thank you!