```
# Databricks notebook source
# MAGIC %md
# MAGIC ## Model Training

# COMMAND ----------

from pyspark.sql.functions import lit
from pyspark.ml.feature import Tokenizer, StopWordsRemover, HashingTF,
IDF
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql.functions import col

# COMMAND ----------

#### Reading, sampling and joining CSVs & setting fake or real label

df_fake = spark.read.csv("/FileStore/tables/fake.csv", header=True,
inferSchema=True)
df_real = spark.read.csv("/FileStore/tables/real.csv", header=True,
inferSchema=True)

#using just 20% sample
df_fake = df_fake.sample(withReplacement=False, fraction=0.2, seed=42)
df_real = df_real.sample(withReplacement=False, fraction=0.2, seed=42)

df_fake = df_fake.withColumn("label", lit(0))
df_real = df_real.withColumn("label", lit(1))
df = df_fake.unionByName(df_real).select("text", "label").na.drop()

df.show(5)

# COMMAND ----------

train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)

# COMMAND ----------

tokenizer = Tokenizer(inputCol='text',outputCol='words')
remover = StopWordsRemover(inputCol='words',outputCol='filtered')
tf = HashingTF(inputCol='filtered', outputCol='rawFeatures')
idf = IDF(inputCol='rawFeatures',outputCol='features')

# COMMAND ----------

rf = RandomForestClassifier(labelCol='label', featuresCol='features',
numTrees=10, maxDepth=4)
pipeline_rf = Pipeline(stages=[tokenizer, remover, tf, idf,rf])

# COMMAND ----------

#### Naive Bayes baseline
```

```
nb = NaiveBayes(labelCol='label', featuresCol='features',
modelType="multinomial")
pipeline_nb = Pipeline(stages=[tokenizer, remover, tf, idf,nb])

# COMMAND ----------

model_rf = pipeline_rf.fit(train_data)
model_nb = pipeline_nb.fit(train_data)

# COMMAND ----------

pred_rf = model_rf.transform(test_data)
pred_nb = model_nb.transform(test_data)

evaluator = BinaryClassificationEvaluator(labelCol="label",
rawPredictionCol="rawPrediction", metricName="areaUnderROC")

auc_rf = evaluator.evaluate(pred_rf)
acc_rf = pred_rf.filter("label = prediction").count() / pred_rf.count()

auc_nb = evaluator.evaluate(pred_nb)
acc_nb = pred_nb.filter("label = prediction").count() / pred_nb.count()

print("Random Forest:")
print(f"AUC: {auc_rf:.4f}")
print(f"Accuracy: {acc_rf:.4f}")
print("\n Naive Bayes:")
print(f"AUC: {auc_nb:.4f}")
print(f"Accuracy: {acc_nb:.4f}")

# COMMAND ----------

model_rf.write().overwrite().save("dbfs:/FileStore/tables/fake_news_best_
model")
print("** Random Forest model saved.")

# COMMAND ----------

# For Random Forest
cm_rf = pred_rf.groupBy("label", "prediction").count().orderBy("label",
"prediction")
cm_rf.show()

# For Naive Bayes
cm_nb = pred_nb.groupBy("label", "prediction").count().orderBy("label",
"prediction")
cm_nb.show()


# COMMAND ----------

from pyspark.sql.functions import udf
from pyspark.ml.linalg import VectorUDT
from pyspark.sql.types import DoubleType
```

```
# Define UDF to extract probability of class 1 (index 1 in vector)
def get_prob1(v):
    return float(v[1])

extract_prob_udf = udf(get_prob1, DoubleType())

# Apply UDF to both models' predictions
rf_points = pred_rf.select("label",
extract_prob_udf("probability").alias("prob1")).toPandas()
nb_points = pred_nb.select("label",
extract_prob_udf("probability").alias("prob1")).toPandas()



# COMMAND ----------

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Random Forest
fpr_rf, tpr_rf, _ = roc_curve(rf_points["label"], rf_points["prob1"])
auc_rf_score = auc(fpr_rf, tpr_rf)

# Naive Bayes
fpr_nb, tpr_nb, _ = roc_curve(nb_points["label"], nb_points["prob1"])
auc_nb_score = auc(fpr_nb, tpr_nb)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(fpr_rf, tpr_rf, label=f"Random Forest (AUC =
{auc_rf_score:.2f})")
plt.plot(fpr_nb, tpr_nb, label=f"Naive Bayes (AUC = {auc_nb_score:.2f})")
plt.plot([0, 1], [0, 1], "k--", label="Random Guessing")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.grid(True)
plt.show()


# COMMAND ----------


dbutils.fs.ls("dbfs:/FileStore/tables/fake_news_best_model")

# COMMAND ----------
```