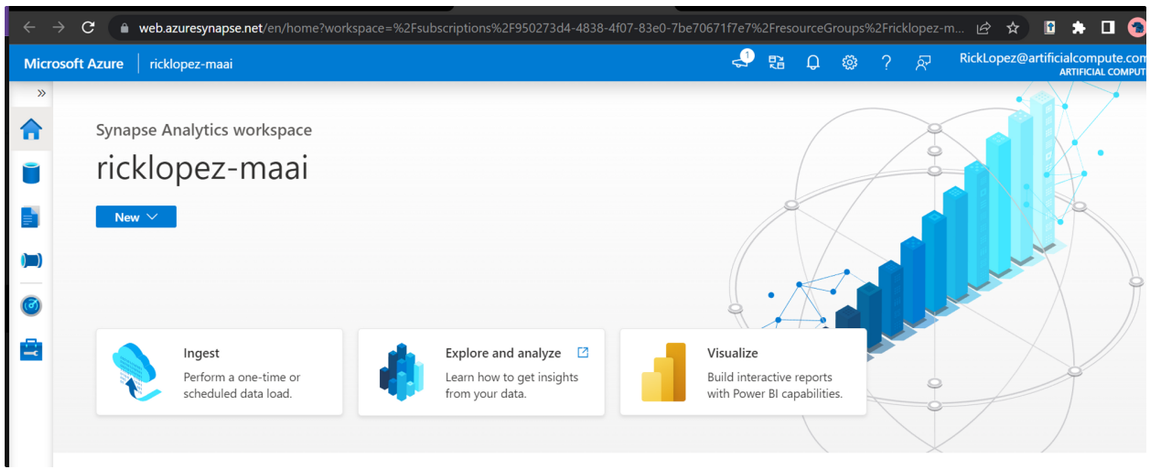
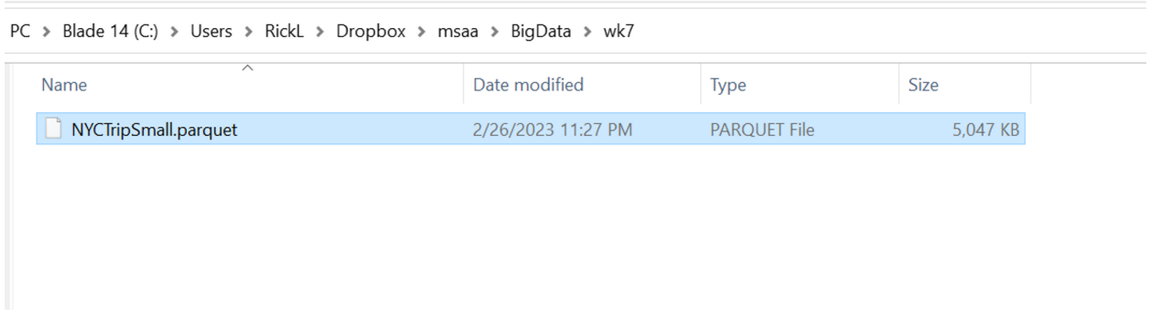
Assignment 7 Screenshots - Richard Lopez

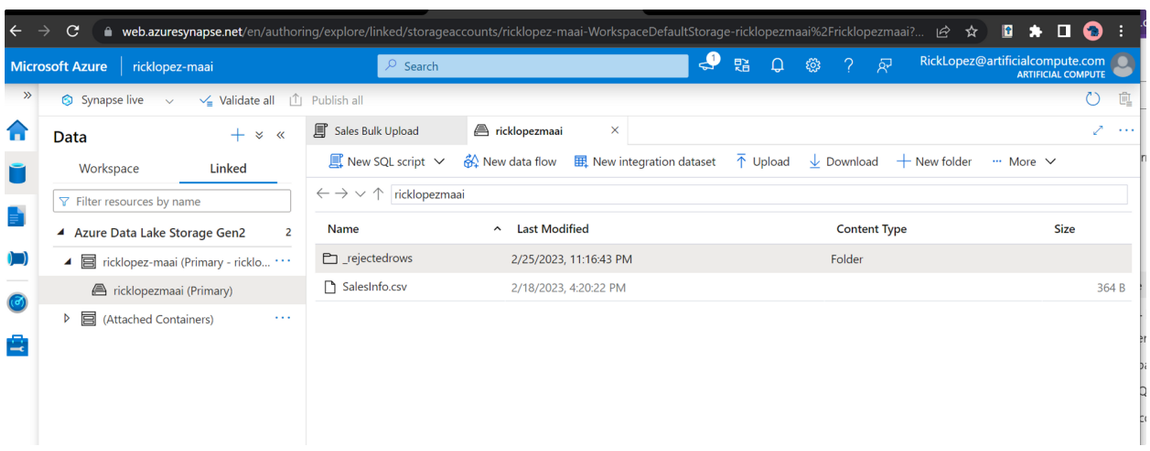
The goal of this assignment is to utilize Azure Synapse integrated Machine Learning capabilities to predict an outcome.

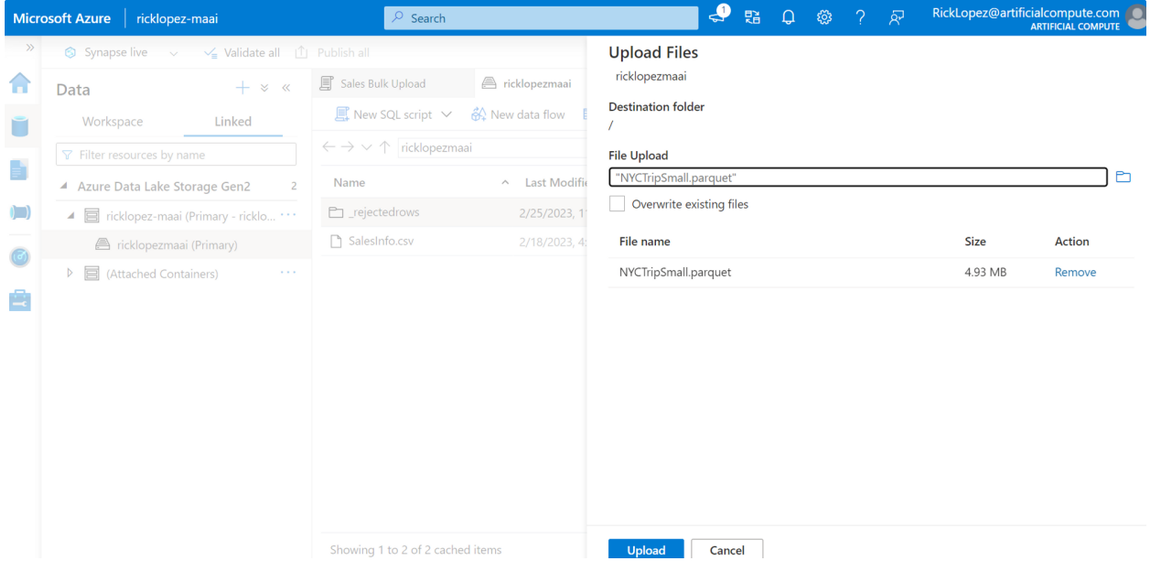


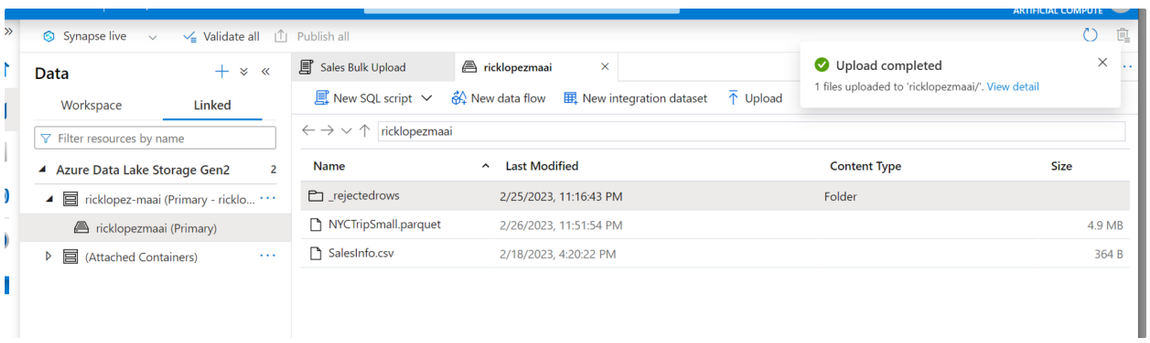
Download and access the dataset

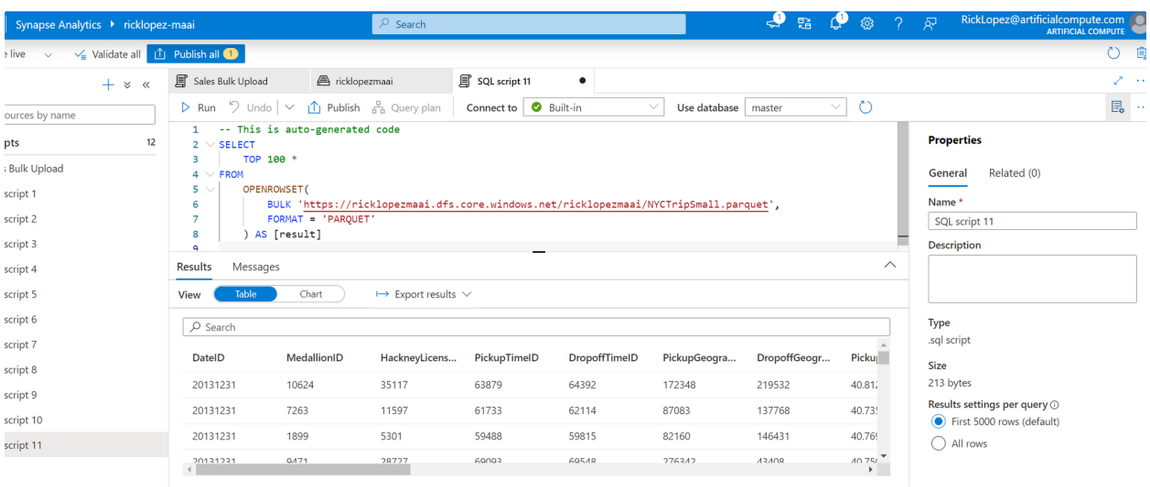


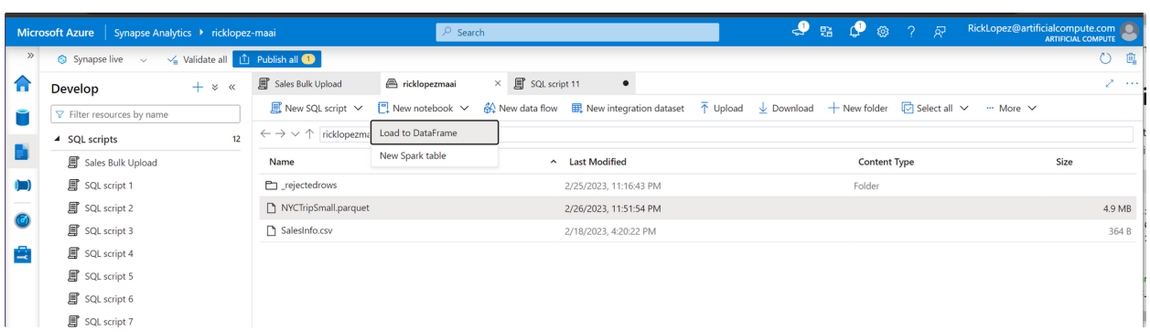
Load the dataset into Azure Synapse

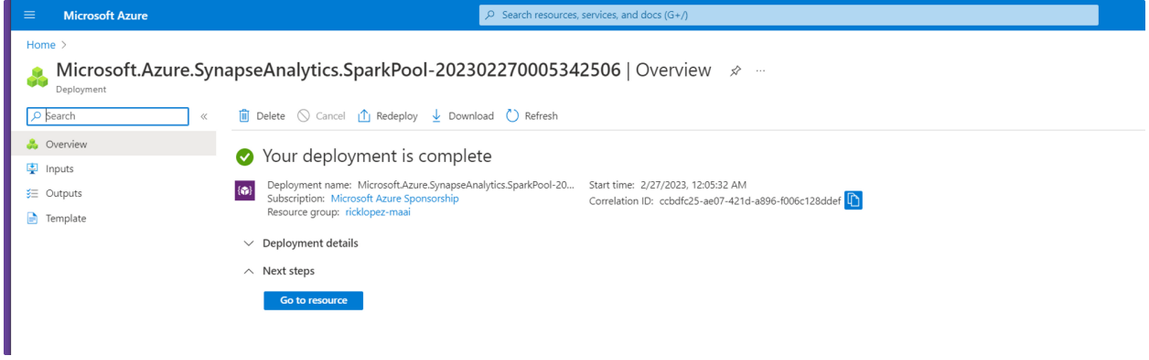


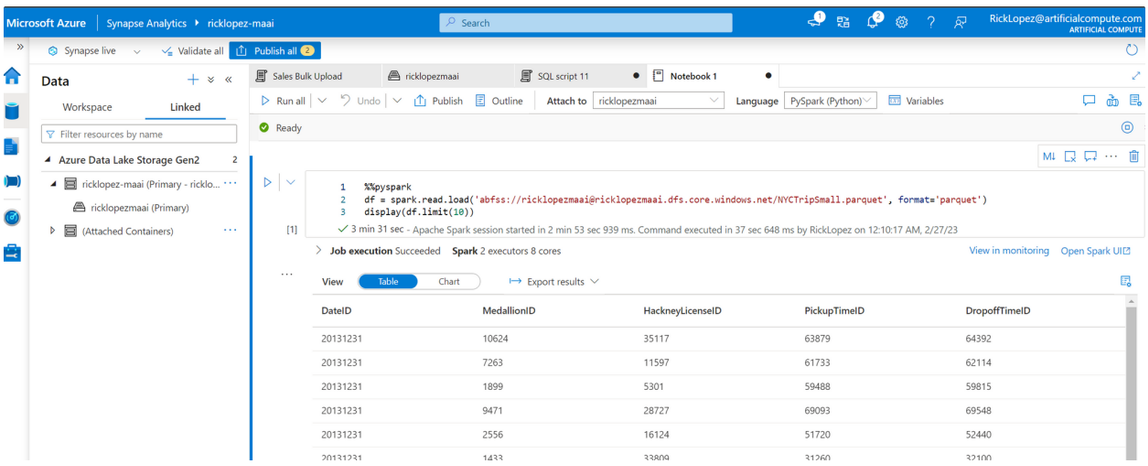

Microsoft Azure
ricklopez-maai
Search
RickLopez@artificialcompute.com
ARTIFICIAL COMPUTE
Synapse live
Validate all
Publish all
Data
Workspace
Linked
Filter resources by name
Azure Data Lake Storage Gen2
ricklopez-maai (Primary - ricklo...
ricklopezmaai (Primary)
(Attached Containers)
Sales Bulk Upload
ricklopezmaai
New SQL script
New data flow
Name
Last Modifi
_rejectedrows
SalesInfo.csv
Showing 1 to 2 of 2 cached items
Upload Files
ricklopezmaai
Destination folder
/
File Upload
"NYCTripSmall.parquet"
Overwrite existing files
File name
Size
Action
NYCTripSmall.parquet
4.93 MB
Remove
Upload
Cancel


Synapse live
Validate all
Publish all
Data
Workspace
Linked
Filter resources by name
Azure Data Lake Storage Gen2
ricklopez-maai (Primary - ricklo...
ricklopezmaai (Primary)
(Attached Containers)
Sales Bulk Upload
ricklopezmaai
New SQL script
New data flow
New integration dataset
Upload
Upload completed
1 files uploaded to 'ricklopezmaai/'. View detail
ricklopezmaai
Name
Last Modified
Content Type
Size
_rejectedrows
2/25/2023, 11:16:43 PM
Folder
NYCTripSmall.parquet
2/26/2023, 11:51:54 PM
4.9 MB
SalesInfo.csv
2/18/2023, 4:20:22 PM
364 B


Synapse Analytics
ricklopez-maai
Search
RickLopez@artificialcompute.com
ARTIFICIAL COMPUTE
Validate all
Publish all
Sales Bulk Upload
ricklopezmaai
SQL script 11
Run
Undo
Publish
Query plan
Connect to
Built-in
Use database
master
1 -- This is auto-generated code
2 SELECT
3 TOP 100 *
4 FROM
5 OPENROWSET(
6 BULK 'https://ricklopezmaai.dfs.core.windows.net/ricklopezmaai/NYCTripSmall.parquet',
7 FORMAT = 'PARQUET'
8) AS [result]
Results
Messages
View
Table
Chart
Export results
Search
DateID
MedallionID
HackneyLicens...
PickupTimeID
DropoffTimeID
PickupGeogra...
DropoffGeogr...
20131231
10624
35117
63879
64392
172348
219532
20131231
7263
11597
61733
62114
87083
137768
20131231
1899
5301
59488
59815
82160
146431
Properties
General
Related (0)
Name *
SQL script 11
Description
Type
.sql script
Size
213 bytes
Results settings per query
First 5000 rows (default)
All rows


Microsoft Azure
Synapse Analytics
ricklopez-maai
Search
RickLopez@artificialcompute.com
ARTIFICIAL COMPUTE
Synapse live
Validate all
Publish all
Develop
Filter resources by name
SQL scripts
Sales Bulk Upload
SQL script 1
SQL script 2
SQL script 3
SQL script 4
SQL script 5
SQL script 6
SQL script 7
Sales Bulk Upload
ricklopezmaai
SQL script 11
New SQL script
New notebook
New data flow
New integration dataset
Upload
Download
New folder
Select all
More
Load to DataFrame
New Spark table
Name
Last Modified
Content Type
Size
_rejectedrows
2/25/2023, 11:16:43 PM
Folder
NYCTripSmall.parquet
2/26/2023, 11:51:54 PM
4.9 MB
SalesInfo.csv
2/18/2023, 4:20:22 PM
364 B





See Full notebook here: https://ricklopez.github.io/AAI-540-01-SP23-Applications-of-AI/wk7/richard-lopez.html

ricklopez.github.io

https://ricklopez.github.io/AAI-540-01-SP23-Applications-of-AI/wk7/richard-lopez.html

Assignment 7.1: Hands-on with Azure Synapse

Richard Lopez

In [1]:
```
%%pyspark
df = spark.read.load('abfss://ricklopezmaai@ricklopezmaai.dfs.core.windows.net/NYCTripSmall.parquet', format='parquet')
display(df.limit(10))
```

```
StatementMeta(ricklopezmaai, 0, 1, Finished, Available)
SynapseWidget(Synapse.DataFrame, dcc47129-9079-4be1-8a5b-ab146bf895fc)
```

Import Libs

In [3]:
```
import matplotlib.pyplot as plt
from datetime import datetime
from dateutil import parser
from pyspark.sql.functions import unix_timestamp, date_format, col, when
from pyspark.ml import Pipeline
from pyspark.ml import PipelineModel
from pyspark.ml.feature import RFormula
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorIndexer
from pyspark.ml.classification import LogisticRegression
from pyspark.mllib.evaluation import BinaryClassificationMetrics
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
from pyspark.mllib.evaluation import BinaryClassificationMetrics
from pyspark.ml.evaluation import BinaryClassificationEvaluator

StatementMeta(ricklopezmaai, 0, 3, Finished, Available)
```

Conduct feature preprocessing (if needed)

[4]:
```
# build DF
from azureml.opendatasets import NycTlcYellow

end_date = parser.parse('2018-06-06')
start_date = parser.parse('2018-05-01')
nyc_tlc = NycTlcYellow(start_date=start_date, end_date=end_date)
filtered_df = nyc_tlc.to_spark_dataframe()
```

```
StatementMeta(ricklopezmaai, 0, 4, Finished, Available)
```

[6]:
```
# Sample data for faster initial development.
sampled_taxi_df = filtered_df.sample(True, 0.002, seed=1234)
```

```
StatementMeta(ricklopezmaai, 0, 6, Finished, Available)
```

[7]:
```
display(sampled_taxi_df)
```

```
StatementMeta(ricklopezmaai, 0, 7, Finished, Available)
SynapseWidget(Synapse.DataFrame, bee7e0ad-d3f5-49cd-8a3d-4d7deaf502c2)
```

Complete Feature Processing

In [8]:
```
taxi_df = sampled_taxi_df.select('totalAmount', 'fareAmount', 'tipAmount', 'paymentType', 'rateCodeId', 'passengerCount'\
                                , 'tripDistance', 'tpepPickupDateTime', 'tpepDropoffDateTime'\
                                , date_format('tpepPickupDateTime', 'hh').alias('pickupHour')\
                                , date_format('tpepPickupDateTime', 'EEEE').alias('weekdayString')\
                                , (unix_timestamp(col('tpepDropoffDateTime')) - unix_timestamp(col('tpepPickupDateTime'))).alias('tripTimeSecs')\
                                , (when(col('tipAmount') > 0, 1).otherwise(0)).alias('tipped')
                                )\
                        .filter((sampled_taxi_df.passengerCount > 0) & (sampled_taxi_df.passengerCount < 8)\
                                & (sampled_taxi_df.tipAmount >= 0) & (sampled_taxi_df.tipAmount <= 25)\
                                & (sampled_taxi_df.fareAmount >= 1) & (sampled_taxi_df.fareAmount <= 250)\
                                & (sampled_taxi_df.tipAmount < sampled_taxi_df.fareAmount)\
                                & (sampled_taxi_df.tripDistance > 0) & (sampled_taxi_df.tripDistance <= 100)\
                                & (sampled_taxi_df.rateCodeId <= 5)
                                & (sampled_taxi_df.paymentType.isin({"1", "2"}))
                                )
```

StatementMeta(ricklopezmaai, 0, 8, Finished, Available)

In [9]:
```
taxi_featurised_df = taxi_df.select('totalAmount', 'fareAmount', 'tipAmount', 'paymentType', 'passengerCount'\
                                        , 'tripDistance', 'weekdayString', 'pickupHour','tripTimeSecs','tipped'\
                                        , when((taxi_df.pickupHour <= 6) | (taxi_df.pickupHour >= 20),"Night")\
                                        .when((taxi_df.pickupHour >= 7) & (taxi_df.pickupHour <= 10), "AMRush")\
                                        .when((taxi_df.pickupHour >= 11) & (taxi_df.pickupHour <= 15), "Afternoon")\
                                        .when((taxi_df.pickupHour >= 16) & (taxi_df.pickupHour <= 19), "PMRush")\
```

In [9]:
```
taxi_featurised_df = taxi_df.select('totalAmount', 'fareAmount', 'tipAmount', 'paymentType', 'passengerCount'\
                                        , 'tripDistance', 'weekdayString', 'pickupHour','tripTimeSecs','tipped'\
                                        , when((taxi_df.pickupHour <= 6) | (taxi_df.pickupHour >= 20),"Night")\
                                        .when((taxi_df.pickupHour >= 7) & (taxi_df.pickupHour <= 10), "AMRush")\
                                        .when((taxi_df.pickupHour >= 11) & (taxi_df.pickupHour <= 15), "Afternoon")\
                                        .when((taxi_df.pickupHour >= 16) & (taxi_df.pickupHour <= 19), "PMRush")\
                                        .otherwise(0).alias('trafficTimeBins')
                                    )\
                                .filter((taxi_df.tripTimeSecs >= 30) & (taxi_df.tripTimeSecs <= 7200))
```

StatementMeta(ricklopezmaai, 0, 9, Finished, Available)

Define the model

In [10]:
```
# convert the categorical columns into numbers
sI1 = StringIndexer(inputCol="trafficTimeBins", outputCol="trafficTimeBinsIndex")
en1 = OneHotEncoder(dropLast=False, inputCol="trafficTimeBinsIndex", outputCol="trafficTimeBinsVec")
sI2 = StringIndexer(inputCol="weekdayString", outputCol="weekdayIndex")
en2 = OneHotEncoder(dropLast=False, inputCol="weekdayIndex", outputCol="weekdayVec")


encoded_final_df = Pipeline(stages=[sI1, en1, sI2, en2]).fit(taxi_featurised_df).transform(taxi_featurised_df)
```

StatementMeta(ricklopezmaai, 0, 10, Finished, Available)

Build the pipeline and train the model

In [11]:
```
# Create training and testing data split from the DataFrame
trainingFraction = 0.7
testingFraction = (1-trainingFraction)
seed = 1234

train_data_df, test_data_df = encoded_final_df.randomSplit([trainingFraction, testingFraction], seed=seed)
```

StatementMeta(ricklopezmaai, 0, 11, Finished, Available)

In [12]:
```
# Logistic regression object for the model
logReg = LogisticRegression(maxIter=10, regParam=0.3, labelCol = 'tipped')

classFormula = RFormula(formula="tipped ~ pickupHour + weekdayVec + passengerCount + tripTimeSecs + tripDistance + fareAmount + paymentType+ trafficT

# training pipeline
lrModel = Pipeline(stages=[classFormula, logReg]).fit(train_data_df)

datestamp = datetime.now().strftime('%m-%d-%Y-%s')
fileName = "lrModel_" + datestamp
logRegDirfilename = fileName
lrModel.save(logRegDirfilename)
```

StatementMeta(ricklopezmaai, 0, 12, Finished, Available)

In [17]:
```
predictions = lrModel.transform(test_data_df)
```

StatementMeta(ricklopezmaai, 0, 17, Finished, Available)

In [19]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator( \
                labelCol='tipped', \
                predictionCol="prediction", \
                metricName="accuracy")
```

StatementMeta(ricklopezmaai, 0, 19, Finished, Available)

In [20]:
```
lr_accuracy = evaluator.evaluate(predictions)
print("Accuracy of LogisticRegression is = %g"%(lr_accuracy))
print("Test Error of LogisticRegression = %g "%(1.0 - lr_accuracy))
```

StatementMeta(ricklopezmaai, 0, 20, Finished, Available)
Accuracy of LogisticRegression is = 0.970428
Test Error of LogisticRegression = 0.0295725

Make predictions and evaluate model performance

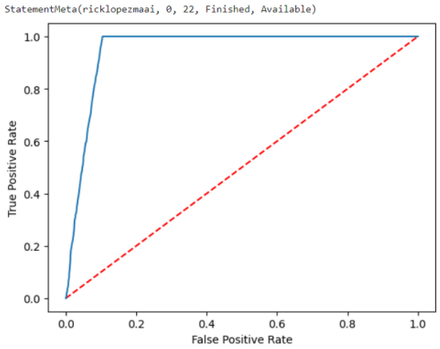
In [21]:
```
predictionAndLabels = predictions.select("label","prediction").rdd
metrics = BinaryClassificationMetrics(predictionAndLabels)
print("Area under ROC = %s" % metrics.areaUnderROC)
```

StatementMeta(ricklopezmaai, 0, 21, Finished, Available)
/opt/spark/python/lib/pyspark.zip/pyspark/sql/context.py:157: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
Area under ROC = 0.9786493385936412

In [22]:
```
## Plot the ROC curve one of the ways to reason about the results of this Model test
modelSummary = lrModel.stages[-1].summary

plt.plot([0, 1], [0, 1], 'r--')
plt.plot(modelSummary.roc.select('FPR').collect(),
         modelSummary.roc.select('TPR').collect())
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```

StatementMeta(ricklopezmaai, 0, 22, Finished, Available)



References

- Chakraborty, P. (2021, January 12). 9 Classification Methods From Spark MLlib We Should Know. Medium; Medium. https://cprosenjit.medium.com/9-classification-methods-from-spark-mllib-we-should-know-c41f555c0425
- AAI-540-01-SP23 - Applications of AI Module 7 Videos (2023). Sandiego.edu. https://ole.sandiego.edu/ultra/courses/_107930_1/cl/outline
- saveenr. (2022, November 18). Quickstart: Get started analyzing with Spark - Azure Synapse Analytics. Microsoft.com. https://learn.microsoft.com/en-us/azure/synapse-analytics/get-started-analyze-spark