Final Project

Reducing Commute Time with Machine Learning and Graph Analysis

Yu, Walter

CSCI E-63 Big Data Analytics

Harvard University Extension School

Prof. Zoran B. Djordjević

Introduction

- Problem: Traffic congestion has negative impacts on roadways, worker productivity and the environment
- Solution: Analyze U.S. commuter data with machine learning (ML) and graph analysis to help reduce commute times
- Github: https://github.com/walteryu/e63-final

Photo Reference: https://lat.ms/2zJGUGZ

NHTS Dataset

- National Household Transportation Survey (NHTS) dataset: https://nhts.ornl.gov/
- National survey with ~1M records; ~700MB total
- Project based on 2017 NHTS Data Challenge contest entry
- ML Analysis: Algorithm selection and feature importance to identify key factors affecting commute times
- Graph Analysis: Evaluate relationships between census division and total vehicle trips

NHTS

U.S. Department of Transportation
Federal Highway Administration

Spark ML Pros/Cons

- Spark ML: https://spark.apache.org/docs/1.2.2/ml-guide.html
- Pros
  - Analyze large datasets quickly
  - Algorithm choices are sufficient for basic analysis
  - Supports data pipelines within Spark/Python environment
- Cons
  - Algorithm choices may be limited for future analysis
  - Data visualization is limited (Matlibplot)

Spark ML Pros/Cons

- GraphX: https://spark.apache.org/docs/1.2.2/graphx-programming-guide.html
- Pros
  - Examine graph relationships between data
  - Supports dataframes; minimal integration required
  - Supports data pipelines within Spark/Python environment
- Cons
  - Not as full-featured as Neo4J or similar graph databases
  - Data visualization is limited (Matlibplot)

Data Analysis

- Data cleaning, exploration and visualization
- Calculate summary statistics
- Analyze with ML and graph analysis
- Document results in Jupyter Notebook

CSCI E-63 Big Data Analytics - Final Project (Fall 2018)

Reducing Commute Time with Machine Learning and Graph Analysis

Author: Walter Yu, Graduate Degree Candidate

Abstract

The average commute time within each U.S. census division has a large impact on its economy, productivity, infrastructure and environment. Longer commute times can lead to lost wages for workers, additional wearing of highway infrastructure and environmental impacts. As a result, this study evaluates U.S. commuter patterns with the National Household Transportation Survey (NHTS) dataset[1] provided by the Federal Highway Administration (FHWA) and whether public transportation or additional transportation planning could reduce commute times based on data analysis.

Data Analysis - Demo

```
# Load trash volume data
hhpub = pd.DataFrame.from_csv('./data/hhpub.csv', index_col=None)
perpub = pd.DataFrame.from_csv('./data/perpub.csv', index_col=None)
trippub = pd.DataFrame.from_csv('./data/trippub.csv', index_col=None)
vehpub = pd.DataFrame.from_csv('./data/vehpub.csv', index_col=None)

# Drop all zero values
hhpub.loc[hhpub.WTHHFIN > 0]
trippub.loc[trippub.WTTRDFIN > 0]

# Remove outliers which are not within 3 standard deviations from mean
hhpub = hhpub[
    np.abs(hhpub.WTHHFIN - hhpub.WTHHFIN.mean()) <= (3*hhpub.WTHHFIN.std())
]
trippub = trippub[
    np.abs(trippub.WTTRDFIN - trippub.WTTRDFIN.mean()) <= (3*trippub.WTTRDFIN.std())
]

# Drop null values since they do not contribute to total
hhpub.dropna(subset=['HOUSEID'], inplace=True)
hhpub.dropna(subset=['HHSTATE'], inplace=True)
hhpub.dropna(subset=['WTHHFIN'], inplace=True)
hhpub.dropna(subset=['CDIVMSAR'], inplace=True)
trippub.dropna(subset=['HOUSEID'], inplace=True)
trippub.dropna(subset=['WTTRDFIN'], inplace=True)
trippub.dropna(subset=['CDIVMSAR'], inplace=True)
perpub.dropna(subset=['CDIVMSAR'], inplace=True)
vehpub.dropna(subset=['ANNMILES'], inplace=True)
```

<<< Code Example
Data Cleaning

Code Example >>>
Data Cleaning

```
# Create bar chart for trips by division
ax = hh_divisions_plot['hh'].plot(
    kind='bar',
    title ="Weighted Value",
    figsize=(12, 6),
    legend=True,
    fontsize=12
)
x_labels = [
    'Mid-Atlantic > 1M with Subway',
    'Mid-Atlantic > 1M w/o Subway',
    'East North Central > 1M with Subway',
    'East North Central > 1M w/o Subway',
    'South Atlantic > 1M with Subway',
    'South Atlantic > 1M w/o Subway',
    'East South Central > 1M with Subway',
    'East South Central > 1M w/o Subway',
    'Pacific > 1M with Subway',
    'Pacific > 1M w/o Subway'
]
plt.title('Weighted Houshold Count by Division', fontsize=16)
ax.set_xlabel("Division, Household Count and Subway System", fontsize=12)
ax.set_ylabel("Weighted Households per Division (Count)", fontsize=12)
ax.set_xticklabels(x_labels)
plt.show()
```

ML and Graph Analysis

- Evaluate ML algorithm performance
- Identify key factors with feature importance
- Develop graph and calculate out-degree relationships

Example ML Output Feature Importance:

prediction	BIKE	BUS	CAR	PARA	PLACE	PRICE	PTRANS	features
1.8759342084160502	5	4	1	5	2	1	4	[5.0,4.0,1.0,5.0,...
2.1851742331474937	5	5	1	5	3	5	5	[5.0,5.0,1.0,5.0,...
2.271102893790943	4	4	1	5	5	5	5	[4.0,4.0,1.0,5.0,...
2.078192432037876	4	5	2	5	3	5	3	[4.0,5.0,2.0,5.0,...
2.348142330357143	2	5	1	5	3	4	5	[2.0,5.0,1.0,5.0,...
2.400746222594453	4	5	1	5	4	3	5	[4.0,5.0,1.0,5.0,...
2.1127218257500227	5	5	1	5	4	4	4	[5.0,5.0,1.0,5.0,...
0.3279613358204647	-9	1	-1	-9	1	1	1	[-9.0,1.0,-1.0,-9...
2.3522649979636734	3	5	1	5	3	2	4	[3.0,5.0,1.0,5.0,...
0.38146184482344175	5	1	4	5	3	5	1	[5.0,1.0,4.0,5.0,...

Example Graph Analysis Output:

```
Total Number of Households: 129696
Total Number of Relationships in Graph: 923572
Total Number of Relationships in Original Data: 923572
```

ML Algorithm Demo - Decision Tree

```
# Per slide 37 of lab 10 notes, prepare data for ML:
vectorAssembler = VectorAssembler(inputCols=['BIKE', 'BUS', 'CAR', 'PARA', 'PLACE', 'PRICE', 'PTRANS'], outputCol='fea
tures')
vhhpub_sp = vectorAssembler.transform(hhpub_sp)
vhhpub_sp

# Per slide 38 of lab 10 notes, split into train/test datasets:
splits = vhhpub_sp.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]

# Per slide 47 of lab 10 notes, prepare data for ML:
dt = DecisionTreeRegressor(featuresCol='features', labelCol='HHVEHCNT')
dt_model = dt.fit(train)

# Per slide 40 of lab 10 notes, describe summary:
# print("DT Model Summary:")
# train.describe().show()

# Per slide 47 of lab 10 notes, create output table:
dt_predictions = dt_model.transform(test)
dt_predictions.select("prediction","BIKE","BUS","CAR","PARA","PLACE","PRICE","PTRANS","features").show(10)

# Per slide 47 of lab 10 notes, evaluate accuracy:
dt_evaluator = RegressionEvaluator(labelCol="HHVEHCNT", predictionCol="prediction", metricName="rmse")
rmse = dt_evaluator.evaluate(dt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
print('')

# Per slide 47 of lab 10 notes, evaluate accuracy:
dt_evaluator = RegressionEvaluator(labelCol="HHVEHCNT", predictionCol="prediction", metricName="r2")
r2 = dt_evaluator.evaluate(dt_predictions)
print("R Squared (R2) on test data = %g" % r2)
print('')

# Print feature importance:
# Reference: https://towardsdatascience.com/building-a-linear-regression-with-pyspark-and-mllib-d065c3ba246a
print('Feature Importance:')
print(dt_model.featureImportances)
```

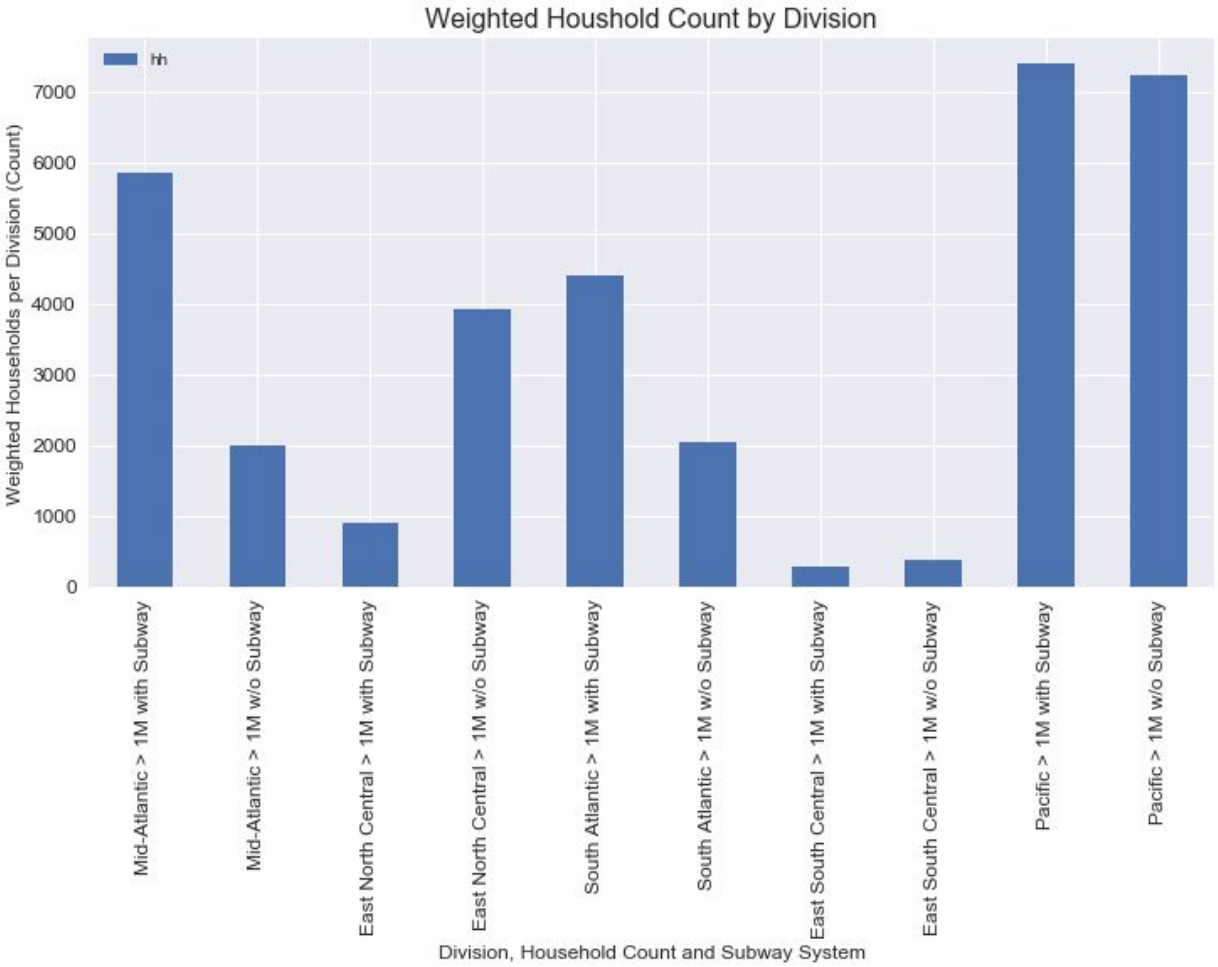
ML Algorithm Demo - Gradient Boosted Tree

```
# Per slide 38 of lab 10 notes, split into train/test datasets:
splits = vhhpub_sp.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1]

# Per slide 47 of lab 10 notes, prepare data for ML:
gbt = GBTRegressor(featuresCol='features', labelCol='HHVEHCNT')
gbt_model = gbt.fit(train)

# Per slide 40 of lab 10 notes, describe summary:
# print("GBT Model Summary:")
# train.describe().show()

# Per slide 47 of lab 10 notes, create output table:
gbt_predictions = gbt_model.transform(test)
gbt_predictions.select("prediction","BIKE","BUS","CAR","PARA","PLACE","PRICE","PTRANS","features").show(10)

# Per slide 47 of lab 10 notes, evaluate accuracy:
gbt_evaluator = RegressionEvaluator(labelCol="HHVEHCNT", predictionCol="prediction", metricName="rmse")
rmse = gbt_evaluator.evaluate(gbt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
print('')

# Per slide 47 of lab 10 notes, evaluate accuracy:
gbt_evaluator = RegressionEvaluator(labelCol="HHVEHCNT", predictionCol="prediction", metricName="r2")
r2 = gbt_evaluator.evaluate(gbt_predictions)
print("R Squared (R2) on test data = %g" % r2)
print('')

# Print feature importance:
# Reference: https://towardsdatascience.com/building-a-linear-regression-with-pyspark-and-mllib-d065c3ba246a
print('Feature Importance:')
print(gbt_model.featureImportances)
```

Graph Analysis Demo

```
# Set vertices/edges:
hh_vertices = hhpub_gx
trip_edges = trippub_gx
hh_vertices = hhpub_gx.withColumnRenamed("CDIVMSAR", "id").distinct()
trip_edges = trippub_gx\
    .withColumnRenamed("CDIVMSAR", "src")\
    .withColumnRenamed("VMT_MILE", "dst")

# Create graph:
# spark = SparkSession.builder.appName(appName).config('spark.jars.packages','graphframes:graphframes:0.6.0-spark2.3-s
_2.11').getOrCreate()
# !pyspark --packages graphframes:graphframes:0.6.0-spark2.3-s_2.11
from graphframes import *
graph = GraphFrame(hh_vertices, trip_edges  )
type(graph)
graph.cache()

# Summary statistics:
print('')
print("Total Number of Households: " + str(graph.vertices.count()))
print("Total Number of Relationships in Graph: " + str(graph.edges.count()))
print("Total Number of Relationships in Original Data: " + str(trip_edges.count()))

# print('')
# print('Show edges:')
# graph.edges.groupBy("src", "dst").count().orderBy(desc("count")).show(50)

# Graph degrees:
print('')
print('Query in-degrees:')
inDeg = graph.inDegrees
inDeg.orderBy(desc("inDegree")).show(50, False)

print('Query out-degrees:')
inDeg = graph.outDegrees
inDeg.orderBy(desc("outDegree")).show(50, False)
```

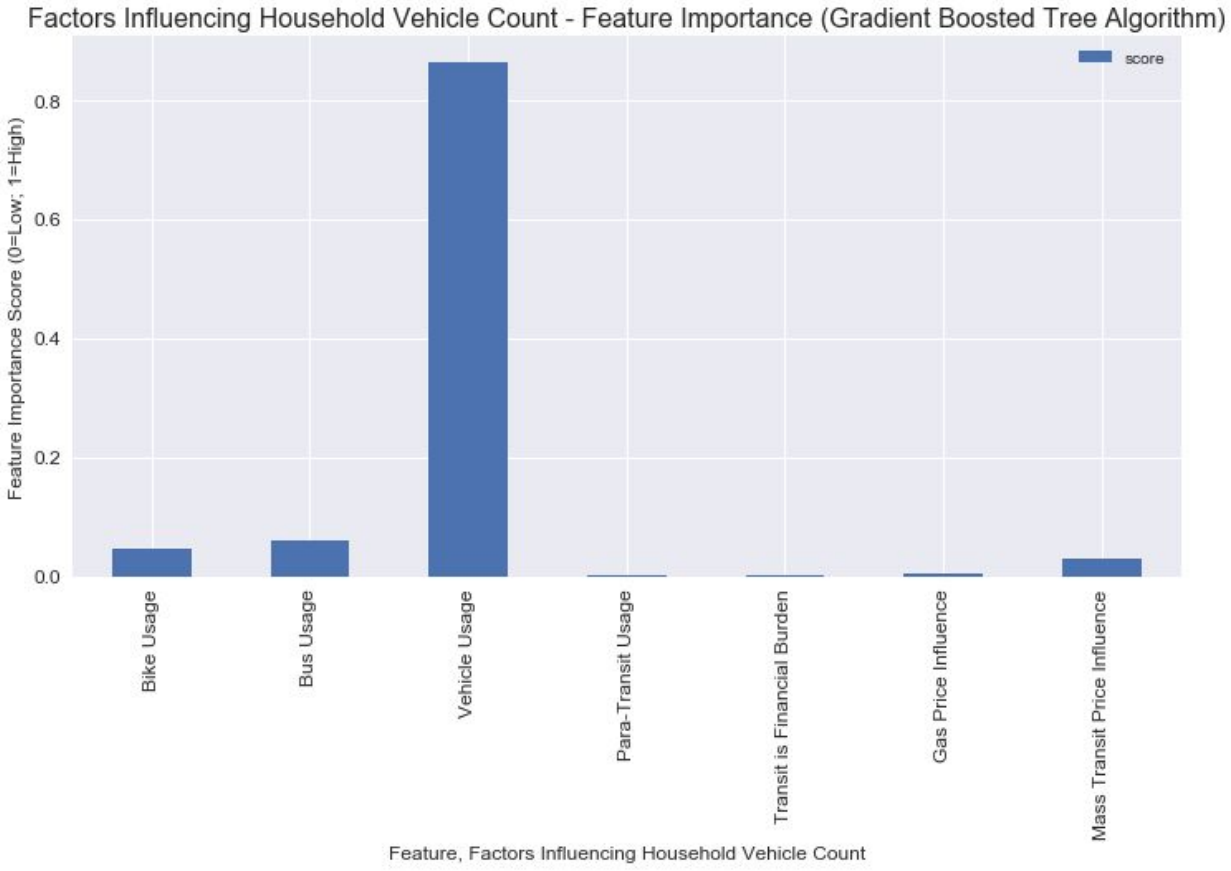
Household Data - Summary Statistics



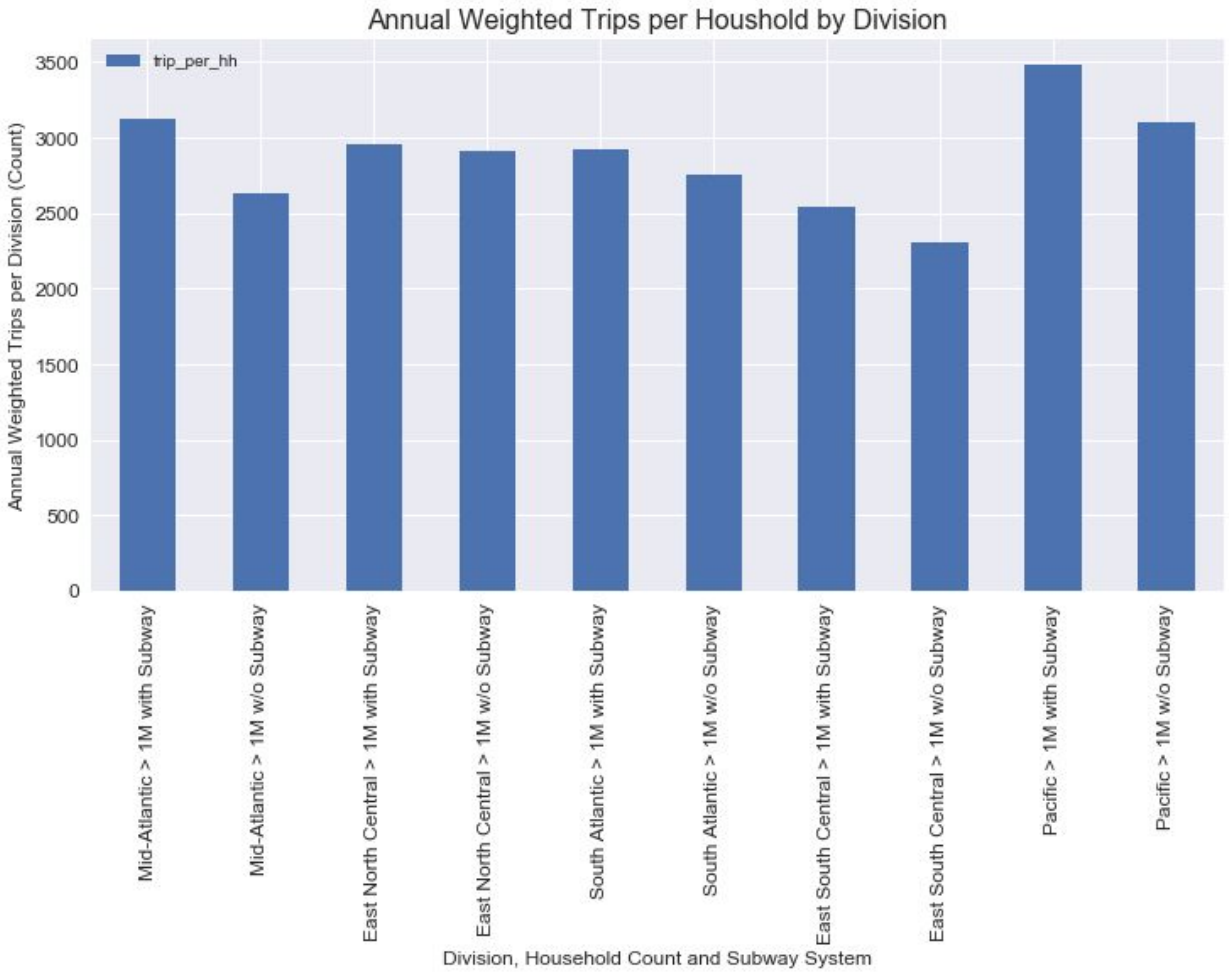
- Calculate household count by division
- Separate by access to mass transit
- Areas with mass transit typically have higher household count

Household Data - Feature Importance

- Vehicle usage most impacts total count
- Other features have low significance
- Mass transit feature has low significance with vehicle usage



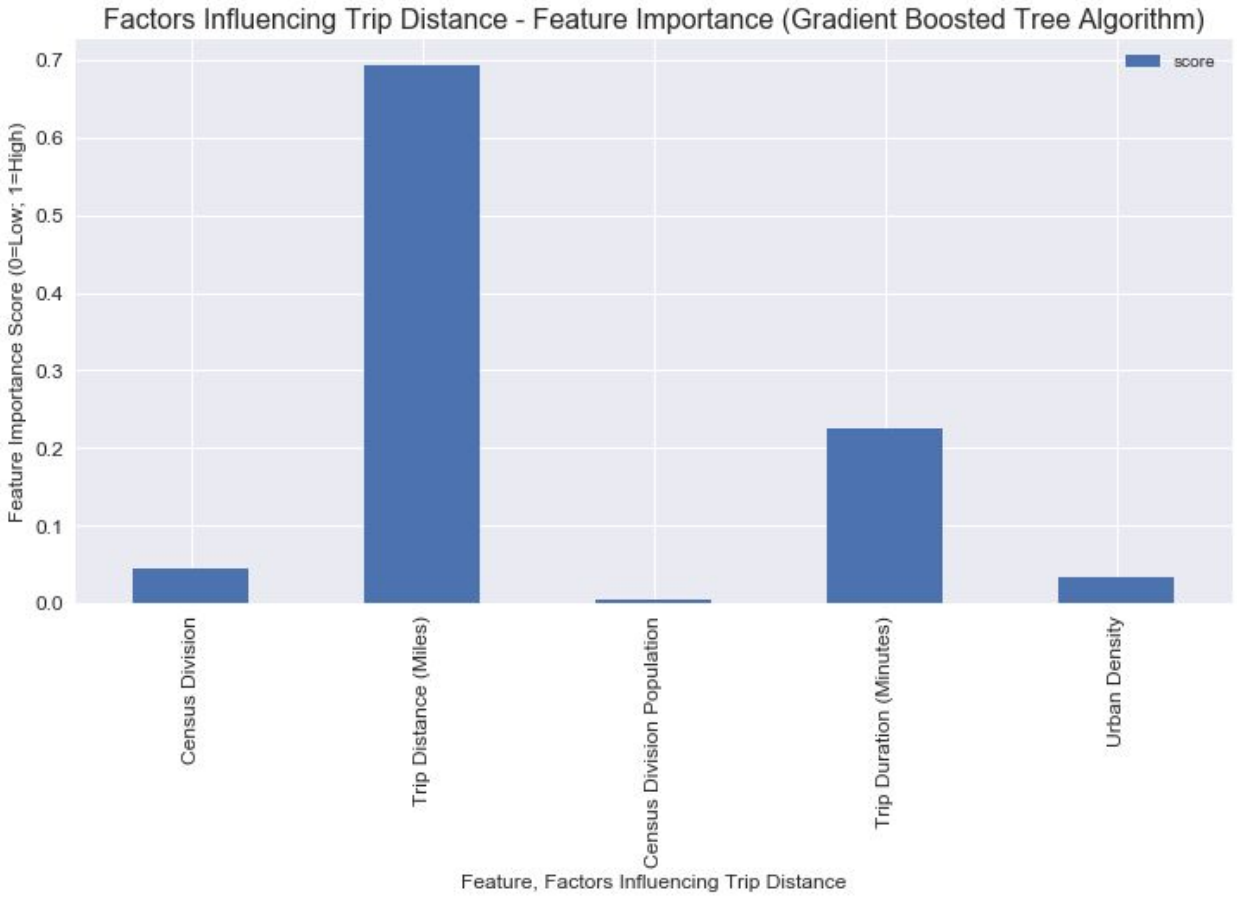
Trip Data - Summary Statistics



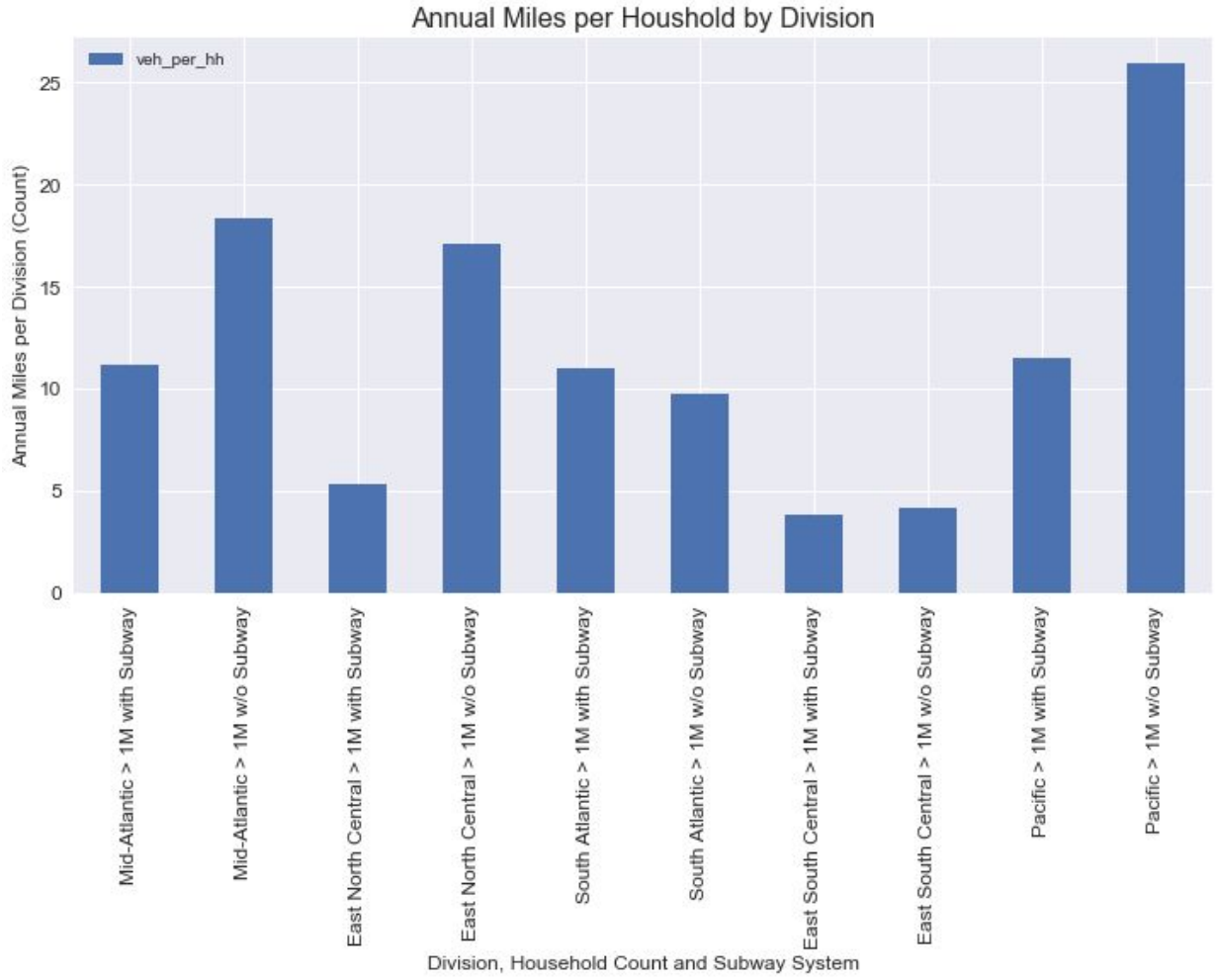
- Calculate trip count by division
- Separate by access to mass transit
- Areas with mass transit typically have higher trip count

Trip Data - Feature Importance

- Trip distance most impacts vehicle miles per trip
- Trip duration is also significant feature
- Other features have low significance

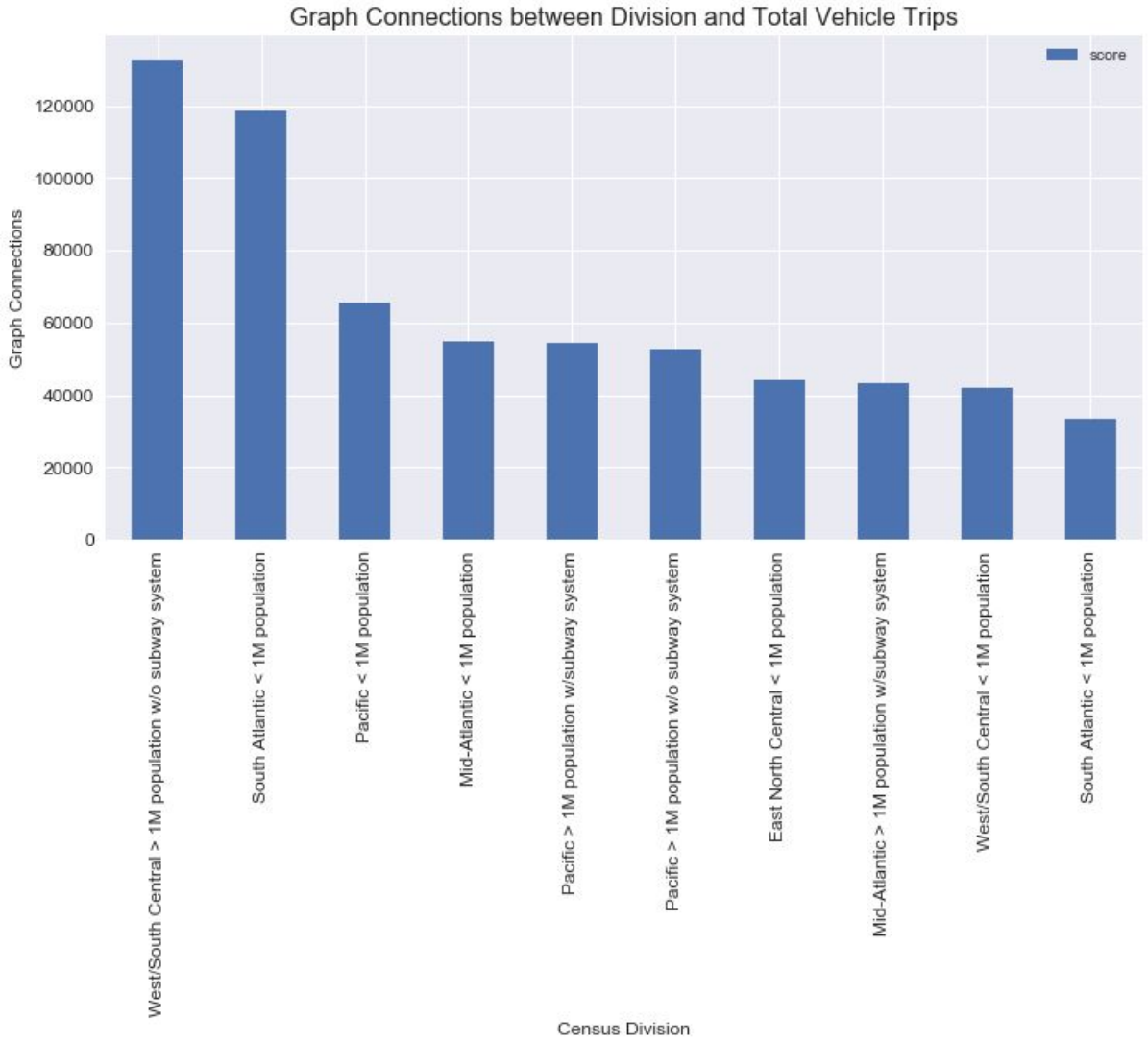


Annual Miles Driven - Summary Statistics



- Calculate annual miles driven by division
- Separate by access to mass transit
- Areas without mass transit typically have higher miles driven

Graph Analysis - Out-Degree Relationship Count



- Calculate out-degree relationships between division and total trips
- Areas without mass transit and lower population

Observations

Photo Reference: http://bit.ly/2DZ2avA

- Mass transit does not appear to directly influence driving behavior; however, trip distance/length has high significance
- Urban areas have higher population so appear to have higher total trips but shorter distances; rural areas have lower population so have lower total trips but longer distances
- Households with higher income appear to have higher vehicle count and usage; trip destination also has significance

Recommendations

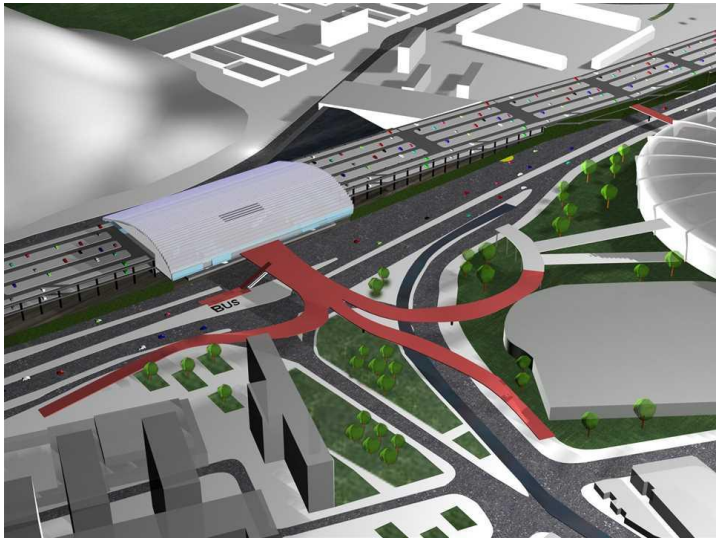


Photo Reference: http://bit.ly/2E2N7Rh

- Public education is recommended to raise awareness
- Results may be useful to urban planners
- Encourage households to take action by living near urban areas instead of relying on urban planning to solve problem
- Mass transit alone is unlikely to reduce vehicle usage; actions to change driving behavior are recommended

YouTube URLs, Last Page

- Two minute (short):
- 15 minutes (long):