Following are three trends from the data:

1. Drivers are concentrated in the urban areas and average fare is lower in the urban areas
2. Drivers are sparse in the rural areas and average fare is higher in the rural areas
3. The above data suggest that if the number of drivers increases, then average fare decreases (i.e Urban) and vice versa (i.e Rural)

Following is the Python/Pandas Code (same as Jupyter notebook). Each section is separated by page

%matplotlib inline

# Dependencies and Setup

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

# File to Load (Remember to change these)

city\_data\_to\_load = "data/city\_data.csv"

ride\_data\_to\_load = "data/ride\_data.csv"

# Read the City and Ride Data

city\_data = pd.read\_csv(city\_data\_to\_load)

ride\_data = pd.read\_csv(ride\_data\_to\_load)

# Combine the data into a single dataset

city\_ride\_data = pd.merge(ride\_data, city\_data, how="left", on="city")

# Display the data table for preview

city\_ride\_data.head()

#group the table by city

city\_ride\_grp\_data = city\_ride\_data.groupby(["city"])

my\_city\_data = city\_data.set\_index("city")

city\_summary\_df = pd.DataFrame({"Fares Per City" : city\_ride\_grp\_data["fare"].mean().map("{:.0f}".format),

"Rides Per City" : city\_ride\_grp\_data["ride\_id"].count(),

"Drivers Per City": my\_city\_data["driver\_count"],

"City Type": my\_city\_data["type"]})

#Separate the data by city types

city\_ride\_urban = city\_summary\_df.loc[city\_summary\_df["City Type"] == "Urban", :].sort\_values("Fares Per City", ascending=True)

city\_ride\_suburban = city\_summary\_df.loc[city\_summary\_df["City Type"] == "Suburban", :].sort\_values("Fares Per City", ascending=True)

city\_ride\_rural = city\_summary\_df.loc[city\_summary\_df["City Type"] == "Rural", :].sort\_values("Fares Per City", ascending=True)

#Plot Urban, multiply the size by 6 to make the boxes look bigger, but proportionate

plt.scatter(city\_ride\_urban["Rides Per City"], city\_ride\_urban["Fares Per City"], marker="o", facecolors="orange", edgecolors="black",

s=city\_ride\_urban["Drivers Per City"]\*7, alpha=0.6, label="Urban")

#Plot Suburban, multiply the size by 6 to make the boxes look bigger, but proportionate

plt.scatter(city\_ride\_suburban["Rides Per City"], city\_ride\_suburban["Fares Per City"], marker="o", facecolors="blue", edgecolors="black",

s=city\_ride\_suburban["Drivers Per City"]\*7, alpha=0.6, label="Suburban")

#Plot Rural, multiply the size by 6 to make the boxes look bigger, but proportionate

plt.scatter(city\_ride\_rural["Rides Per City"], city\_ride\_rural["Fares Per City"], marker="o", facecolors="yellow", edgecolors="black",

s=city\_ride\_rural["Drivers Per City"]\*7, alpha=0.6, label="Rural")

x\_max = city\_summary\_df["Rides Per City"].max() + 2

y\_max = int(city\_summary\_df["Fares Per City"].max()) - int(city\_summary\_df["Fares Per City"].min())

# set xlim and ylim

plt.xlim(0, x\_max)

plt.ylim(-2, y\_max)

plt.yticks(np.arange(0, y\_max, 5))

# set the legent

lgnd = plt.legend()

# make the legend bubbles the same size

for handle in lgnd.legendHandles:

handle.set\_sizes([60])

# Add labels to the x and y axes

plt.title("Pyber Ride Sharing Data (2016)")

plt.xlabel("Total Number of Rides (Per City)")

plt.ylabel("Average Fare ($)")

# Create a legend

# Set a grid on the plot

plt.grid()

plt.tight\_layout()

# Save Figure

plt.savefig("../Images/AshokConfigScatter.png")

# Calculate Type Percents

all\_city\_fare = city\_ride\_data["fare"].sum()

urban\_city\_fare = city\_ride\_data.loc[city\_ride\_data["type"] == "Urban", :]["fare"].sum()

urban\_city\_fare\_percent = urban\_city\_fare / all\_city\_fare

suburban\_city\_fare = city\_ride\_data.loc[city\_ride\_data["type"] == "Suburban", :]["fare"].sum()

suburban\_city\_fare\_percent = suburban\_city\_fare / all\_city\_fare

rural\_city\_fare = city\_ride\_data.loc[city\_ride\_data["type"] == "Rural", :]["fare"].sum()

rural\_city\_fare\_percent = rural\_city\_fare / all\_city\_fare

# Build Pie Chart

# Labels for the sections of our pie chart

labels = ["Urban", "Rural", "Suburban"]

# The values of each section of the pie chart

sizes = [urban\_city\_fare\_percent, rural\_city\_fare\_percent, suburban\_city\_fare\_percent]

# The colors of each section of the pie chart

colors = ["lightcoral", "yellow", "lightskyblue"]

# Tells matplotlib to seperate the "Python" section from the others

explode = (0.08, 0, 0)

# Creates the pie chart based upon the values above

# Automatically finds the percentages of each part of the pie chart

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct="%1.1f%%", shadow=True, startangle=270)

plt.axis("equal")

# Save Figure

plt.title("% of Total Fares by City Type")

# Save Figure

plt.savefig("../Images/AshokConfigPie1.png")

plt.show()

# Calculate Ride Percents

all\_ride\_count = city\_ride\_data["ride\_id"].count()

urban\_ride\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Urban", :]["ride\_id"].count()

urban\_ride\_percent = urban\_ride\_count / all\_ride\_count

suburban\_ride\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Suburban", :]["ride\_id"].count()

suburban\_ride\_percent = suburban\_ride\_count / all\_ride\_count

rural\_ride\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Rural", :]["ride\_id"].count()

rural\_ride\_percent = rural\_ride\_count / all\_ride\_count

# Build Pie Chart

# Labels for the sections of our pie chart

labels = ["Urban", "Rural", "Suburban"]

# The values of each section of the pie chart

sizes = [urban\_ride\_percent, rural\_ride\_percent, suburban\_ride\_percent]

# The colors of each section of the pie chart

colors = ["lightcoral", "yellow", "lightskyblue"]

# Tells matplotlib to seperate the "Python" section from the others

explode = (0.08, 0, 0)

# Creates the pie chart based upon the values above

# Automatically finds the percentages of each part of the pie chart

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct="%1.1f%%", shadow=True, startangle=270)

plt.axis("equal")

# Save Figure

plt.title("% of Total Rides by City Type")

# Save Figure

plt.savefig("../Images/AshokConfigPie2.png")

plt.show()

# Calculate Driver Percents

all\_driver\_count = city\_ride\_data["driver\_count"].sum()

urban\_driver\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Urban", :]["driver\_count"].sum()

urban\_driver\_percent = urban\_driver\_count / all\_driver\_count

suburban\_driver\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Suburban", :]["driver\_count"].sum()

suburban\_driver\_percent = suburban\_driver\_count / all\_driver\_count

rural\_driver\_count = city\_ride\_data.loc[city\_ride\_data["type"] == "Rural", :]["driver\_count"].sum()

rural\_driver\_percent = rural\_driver\_count / all\_driver\_count

# Build Pie Chart

# Labels for the sections of our pie chart

labels = ["Urban", "Rural", "Suburban"]

# The values of each section of the pie chart

sizes = [urban\_driver\_percent, rural\_driver\_percent, suburban\_driver\_percent]

# The colors of each section of the pie chart

colors = ["lightcoral", "yellow", "lightskyblue"]

# Tells matplotlib to seperate the "Python" section from the others

explode = (0, 0.1, 0.1)

# Creates the pie chart based upon the values above

# Automatically finds the percentages of each part of the pie chart

plt.pie(sizes, explode=explode, labels=labels, colors=colors,

autopct="%1.1f%%", shadow=True, startangle=270)

plt.axis("equal")

# Save Figure

plt.title("% of Total Drivers by City Type")

# Save Figure

plt.savefig("../Images/AshokConfigPie3.png")

plt.show()