Exploring Relationships Between Transport Investment and Passenger Ridership

SI618 Final Project - Winter 2023
Tyler Allen, Elaine Czarnik, and Eman Mozaffar

Motivation

Every country allocates its transportation funding in different ways, but does this have an effect on how people choose to get around? Does increasing funding in one area improve usage, or is it not worth investing after a certain point? To answer these questions and understand general transportation-related trends across countries, we sought to clean, manipulate, analyze and predict on data provided by sources linked to the European Union.

We want to study this data so we can understand how investment in multimodal transportation affects usage of railways, cars, and other modes of transport, and to uncover any other trends from the early 2000s until now. Transportation has many cascading effects on areas such as quality of life, infrastructure, happiness, environmental sustainability, and social habits. The way we get around is central to how we live our lives. In addition, we focused on the European Union to maintain consistency and availability.

Data Sources

Our primary dataset captures the number of passenger cars per 1000 inhabitants from Eurostat. The dataset features passenger car inhabitant data across 40 different European countries from 2012 to 2022, and contains 400 records. Three secondary datasets were also derived from the Eurostat database, focusing on the number of passengers transported by railways, population summary statistics, and GDP per capita by country. Each of these datasets had 360, 320, and 540 records, respectively. The Eurostat datasets give us an overview of the number of cars used by each country, which is an indicator of car dependency, and the number of passengers who are regularly using railways, which is an indicator of public transportation usage.

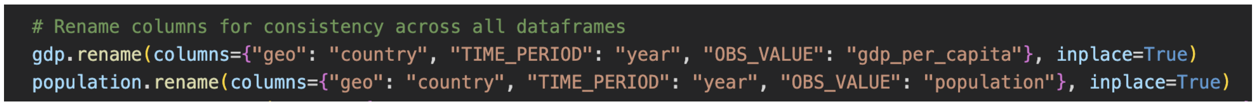
We also gathered data from the International Transport Forum. This data captures total road spending, along with overall transportation infrastructure investment and spending from 2000 to 2020. Road spending covers all money the country invested in road maintenance, highway building, and other car-dependent modes of transport. Overall transportation infrastructure investment focuses on road spending, in addition to public transportation, railways, and local maritime spending, if applicable.

We were able to pull this data directly from the Eurostat and ITF websites, respectively, in CSV format. All of these data sources were recorded by country and year, which allowed us to combine our data on those fields.

Data Manipulation Methods

Data Cleansing

All 5 CSV files were loaded into the notebook and saved as pandas dataframes. Upon reviewing the head of each dataframe, we identified several columns that would not be needed for our analysis (such as when the data was last updated) and dropped them. Following the removal of extraneous columns, we renamed some of the remaining column names to maintain consistency across each dataframe (e.g., renamed 'geo' columns to 'country'), which also helped set them up for joining later on.



```
# Rename columns for consistency across all dataframes
gdp.rename(columns={"geo": "country", "TIME_PERIOD": "year", "OBS_VALUE": "gdp_per_capita"}, inplace=True)
population.rename(columns={"geo": "country", "TIME_PERIOD": "year", "OBS_VALUE": "population"}, inplace=True)
```

Dataframe Specific Cleansing

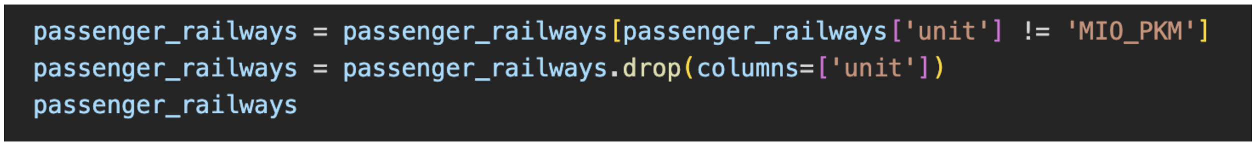
Transportation Investment Dataframe

For the transportation_investment data frame, the investment 'measure' was inconsistent across countries, and some countries had multiple entries in Euros and their local currency (e.g., Bulgaria had duplicate rows of investment data measured in Euros and Bulgarian Lev). We filtered the investment 'measure' to be just in Euros to maintain consistency. We also filtered the dataframe to only include results for 'total inland transport infrastructure investment' and 'total road spending' for each country, as these are the primary variables we want for our analysis.

The transportation_investment data frame was later split into two separate dataframes: one for 'total inland transport infrastructure investment' and the second for 'total road spending'. Splitting the initial transportation_investment data frame into two separate tables was necessary so we could successfully join our tables.

Passenger Railways Dataframe

For the passenger_railways dataframe, the 'passengers_transported' column initially had duplicate rows of data for each country to show the number of passengers transported in thousands of passengers per kilometer (THS_PKM) and millions of passengers per kilometer (MIO_PKM). We filtered out the rows of data that included millions of passengers per kilometer so our resulting passenger_railways data frame only included the columns 'country', 'year', and 'passengers_transported' measured in thousands per kilometer.



```
passenger_railways = passenger_railways[passenger_railways['unit'] != 'MIO_PKM']
passenger_railways = passenger_railways.drop(columns=['unit'])
passenger_railways
```

Changing Country Codes

The final cleansing applied to each dataframe was changing the 'country' column codes to the country's full name. This was done by looping through each dataframe's 'country' column and updating the country's code to its full name if there was a match.

Joining Dataframes

Following data cleaning, all six dataframes were joined together into one. We performed the joins on two tables at a time, specifically performing a left join on the columns sharing a ‘country’ and ‘year’. The resulting merged dataframe had 360 rows of data and 8 columns.

```
merged_df1 = passenger_cars.merge(passenger_railways, on=['country', 'year'], how='left')
merged_df1
```

Null Values

We identified four columns that had several rows of null values: ‘passengers_transported’ had 70 null values; ‘gdp_per_capita’ had 13 null values; ‘total_road_spending’ had 160 null values; ‘total_inland_transport_invesment’ had 130 null values. Some of the countries in our final dataframe did not have values for some columns, or had data for a few years but not all. We decided to drop these rows from the dataframe, as it would be difficult to analyze if maintained. It would also require significant research for each country and variable to identify what the best value would be to fill in. As a result, the final dataframe had 150 rows of data for 20 countries.

Feature Engineering

Region

Before beginning our analyses, we created new features that would be important for our clustering and linear regression analyses. We added a ‘region’ feature where each country was mapped to its corresponding European region as defined by EuroVoc (e.g., the country Austria was mapped to the ‘western_europe’ region). The new ‘region’ column includes four distinct regions: northern_europe, western_europe, souther_europe, and central_eastern_europe.

Population Size

Additionally, another feature called ‘pop_size’ was also added. The original ‘population’ data was mapped to a corresponding population size category based on its value. The new ‘pop_size’ column includes three distinct population size categories: ‘Small’ for populations with a value less than 5,000,000; ‘Medium’ for populations with a value between 5,000,000 and 20,000,000, inclusive; ‘Large’ for populations with a value greater than 20,000,000.

Spending and Passengers Transported

For the remaining spending and passengers_transported features, new features were created for each by transforming their measures to ‘per 1000’. This transformation was applied to create normalization across the features in our data set. The following features were transformed to ‘per 1000’: ‘passengers_transported’, ‘total_inland_transport_investment’, and ‘total_road_spending’.

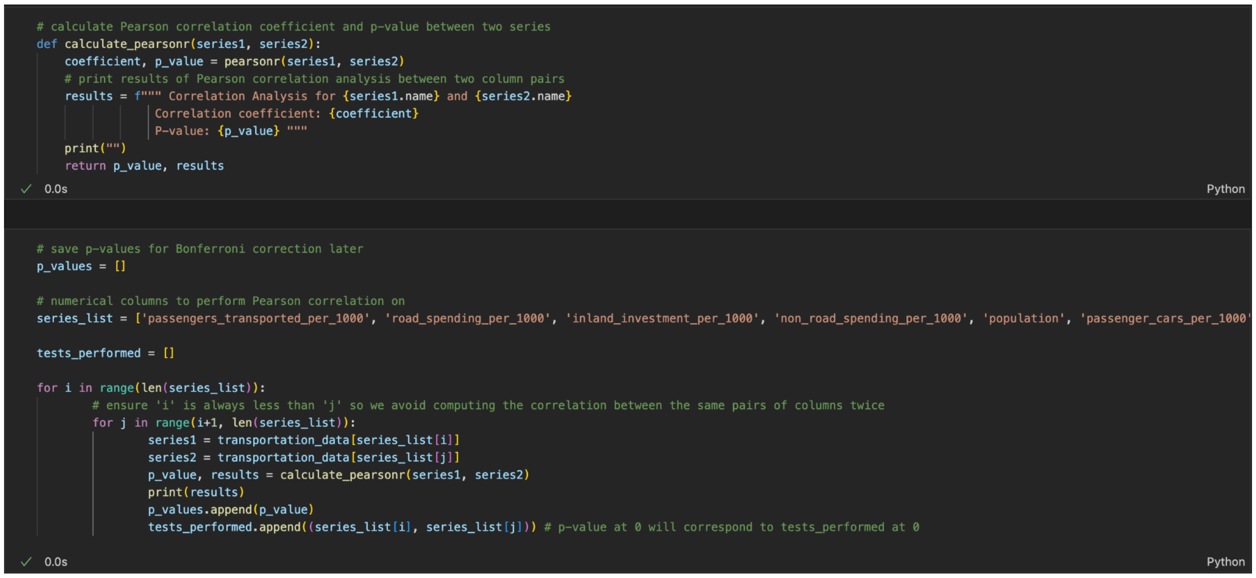
Additionally, a new feature called ‘non_road_spending’ was added by subtracting ‘total_inland_transport_investment’ from ‘total_road_spending’. Afterward, the same ‘per 1000’ transformation was applied to ‘non_road_spending’ to ensure uniformity across our features.

Analysis

Correlation Analysis

Initially, we calculated simple pairwise correlations between all numerical variables in our dataset. We calculated the Pearson correlation coefficient to understand the strength and direction of the linear relationship between each pair of numerical variables. The Pearson correlation coefficient helps us identify pairs of variables that have a positive relationship (coefficients with a value between 0 and 1), a negative relationship (coefficients with a value between 0 and -1), or no relationship (coefficients with a value of 0). We used the pearsonr function from the scipy.stats library to calculate the correlation coefficients.

Additionally, the pearsonr function also calculates the p-value between two variables. The p-value returned tells us the probability of obtaining the computed Pearson correlation coefficient if the null hypothesis states there is no relationship between the two variables. For our test, we will assume an initial alpha of 0.05. If our p-values are less than 0.05, we can reject the null hypothesis and conclude there is evidence of a statistically significant correlation between two variables.



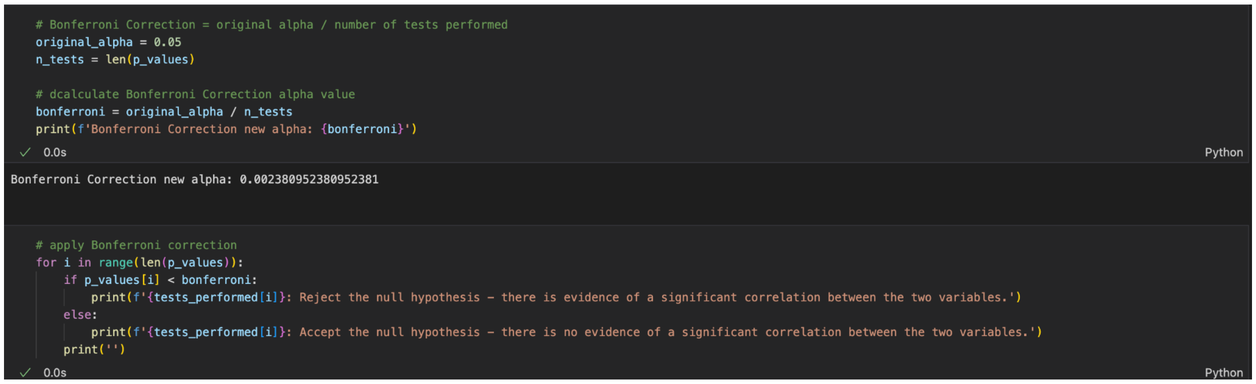
```python
# calculate Pearson correlation coefficient and p-value between two series
def calculate_pearsonr(series1, series2):
    coefficient, p_value = pearsonr(series1, series2)
    # print results of Pearson correlation analysis between two column pairs
    results = f""" Correlation Analysis for {series1.name} and {series2.name}
                Correlation coefficient: {coefficient}
                P-value: {p_value} """
    print("")
    return p_value, results
```
✓ 0.0s Python

```python
# save p-values for Bonferroni correction later
p_values = []

# numerical columns to perform Pearson correlation on
series_list = ['passengers_transported_per_1000', 'road_spending_per_1000', 'inland_investment_per_1000', 'non_road_spending_per_1000', 'population', 'passenger_cars_per_1000'

tests_performed = []

for i in range(len(series_list)):
        # ensure 'i' is always less than 'j' so we avoid computing the correlation between the same pairs of columns twice
        for j in range(i+1, len(series_list)):
                series1 = transportation_data[series_list[i]]
                series2 = transportation_data[series_list[j]]
                p_value, results = calculate_pearsonr(series1, series2)
                print(results)
                p_values.append(p_value)
                tests_performed.append((series_list[i], series_list[j])) # p-value at 0 will correspond to tests_performed at 0
```
✓ 0.0s Python

After calculating our Pearson correlation coefficient for each of our numerical variable pairs, we can see all but two of our pairs have a positive correlation coefficient, and therefore have a positive relationship with one another. The pair 'road_spending_per_1000' and 'population', and the pair 'population' and 'gdp_per_capita' have negative correlation coefficients, indicating a negative correlation with one another.

However, are any of these relationships significant? We can look at our resulting p-values for each test and compare them to our alpha value to determine if the relationship is statistically significant. But before we can do that, we need to perform a Bonferroni Correction to reduce the chance of obtaining several false-positive results. We will obtain a new alpha to which we can compare our p-values so we can determine if there is a truly significant relationship.

To calculate the Bonferroni Correction, we simply divide our initial alpha value (0.05) by the total number of pairwise tests performed (n=21). The new alpha value is now 0.002. Therefore, for any pairs with a p-value less the 0.002 we will reject the null hypothesis and conclude there is a statistically significant correlation between the two variables; for pairs with a p-value greater than 0.002, we will accept the null hypothesis and conclude there is no statistically significant correlation between the two variables.



```python
# Bonferroni Correction = original alpha / number of tests performed
original_alpha = 0.05
n_tests = len(p_values)

# dcalculate Bonferroni Correction alpha value
bonferroni = original_alpha / n_tests
print(f'Bonferroni Correction new alpha: {bonferroni}')
```
✓ 0.0s Python

```
Bonferroni Correction new alpha: 0.002380952380952381
```

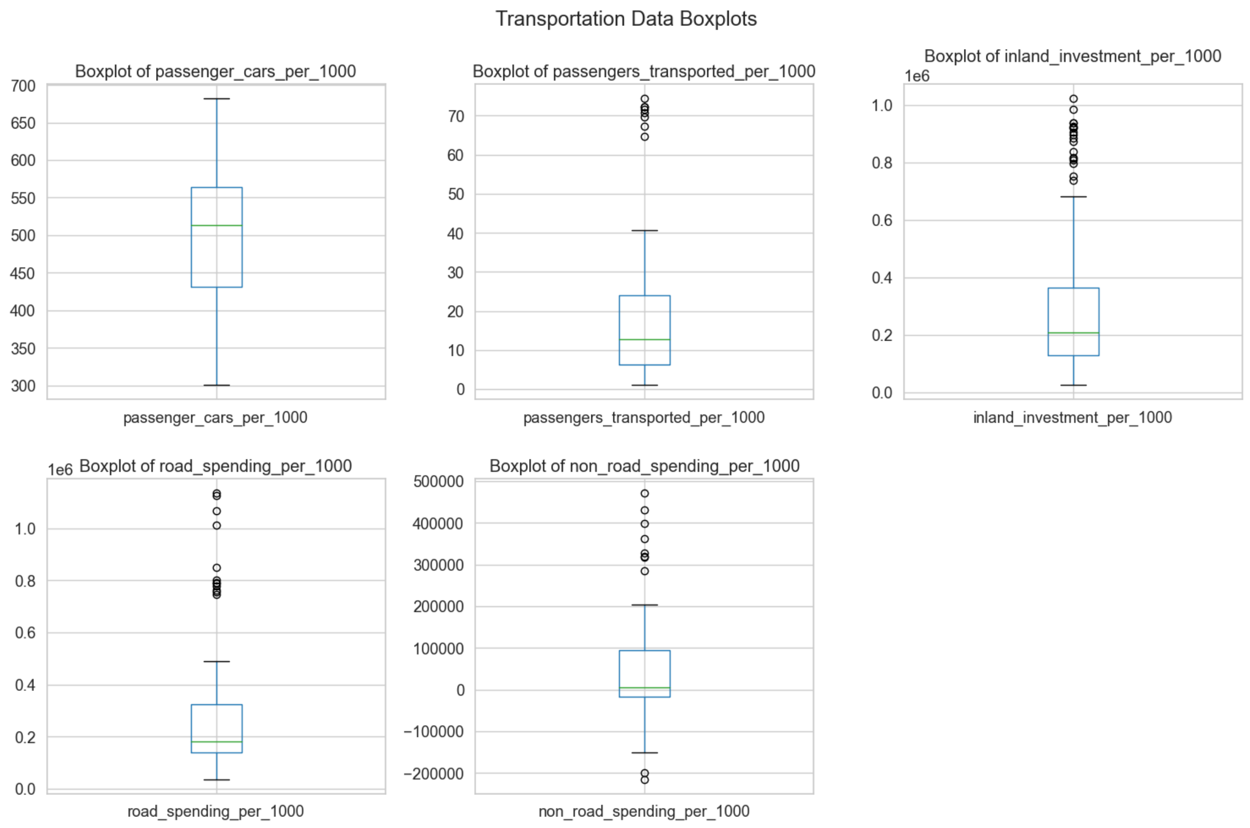
```python
# apply Bonferroni correction
for i in range(len(p_values)):
    if p_values[i] < bonferroni:
        print(f'{tests_performed[i]}: Reject the null hypothesis - there is evidence of a significant correlation between the two variables.')
    else:
        print(f'{tests_performed[i]}: Accept the null hypothesis - there is no evidence of a significant correlation between the two variables.')
    print('')
```
✓ 0.0s Python

After applying the Bonferroni Correction, we can conclude there are 15 pairs that show a statistically significant correlation, and 6 that do not. Most notably, there is a significant relationship between the number of passengers transported per 1000 and the type of investment or spending a country puts into its roads, non-roads, and other infrastructure. We will explore this further to better understand the relationship between the passengers transported and investment features.

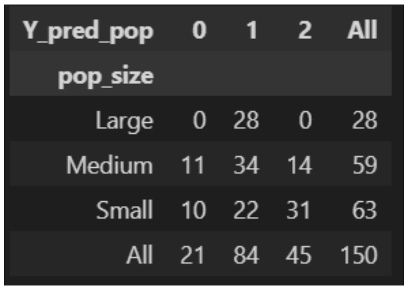
Clustering

Knowing that there is a relationship within our data, the question then turned to whether we could learn anything from clustering. We were curious to know if there were clusters for population size, region, and country using the per 1000 population variables for passenger cars, passengers transported on railway, inland investment, road spending, and non-road spending. We created one feature matrix of these per 1000 population variables, and created a pipeline for population size, region, and country, where the number of clusters matched the number of categories for each variable: three for population size, four for region, and 20 for country.

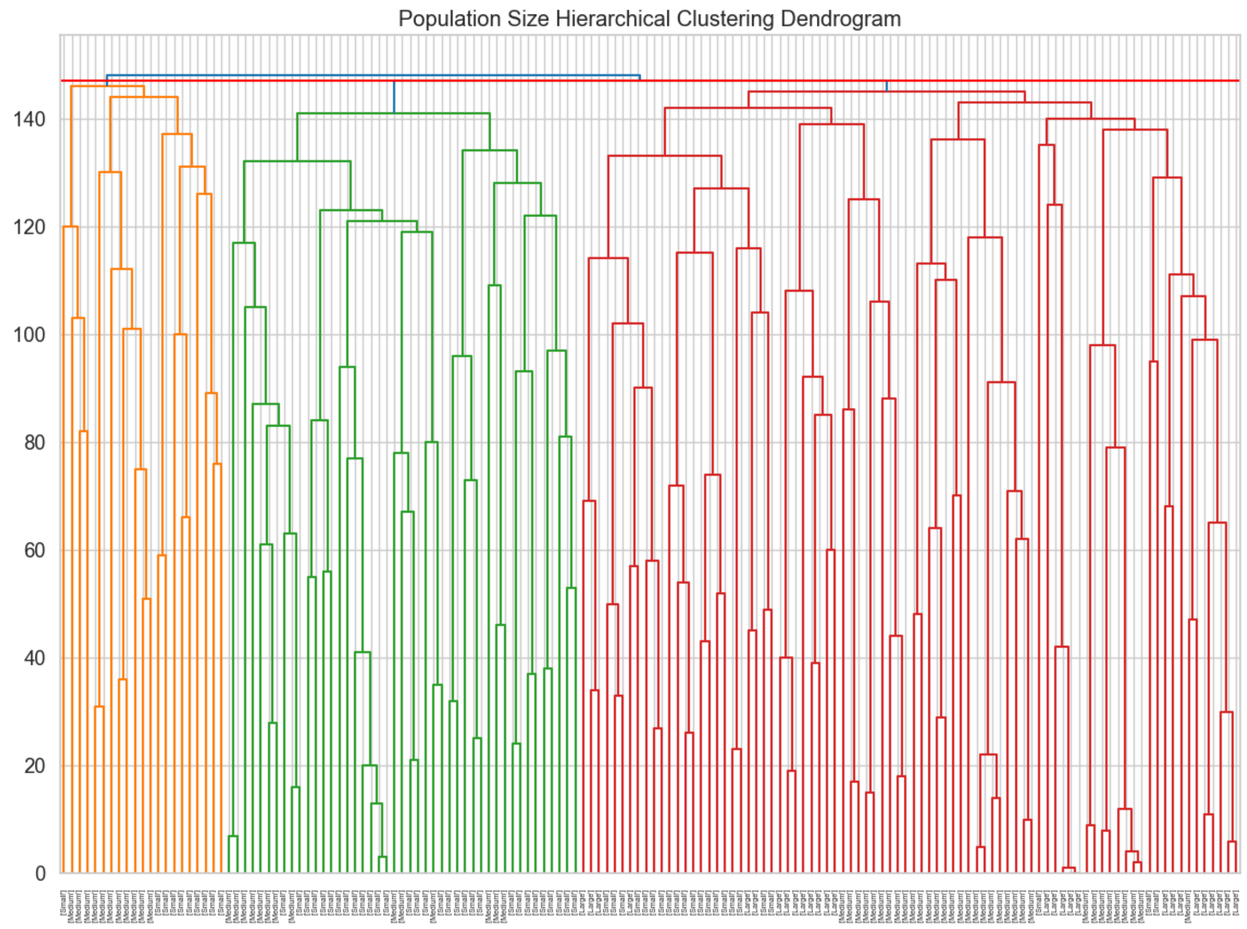
We chose to use agglomerative clustering in the pipeline because there were no clearly defined clusters in any of the variables, and after creating boxplots, we found outliers for multiple features. This makes agglomerative clustering a better choice than k-means clustering.



After making the clusters we then created dendrograms, crosstabs, and pairplots to view the clustering results. We'll talk about population size in this report, but the same approach was taken for region and country. We started by creating a dendrogram of the clustering results and setting the visualization to show three clusters to match the small, medium, and large population sizes. We then created a crosstab to view the original variable compared to the clustering results.



Y_pred_pop	0	1	2	All
pop_size				
Large	0	28	0	28
Medium	11	34	14	59
Small	10	22	31	63
All	21	84	45	150



The crosstab tells us that there should be a cluster that is much smaller than the other two, but instead we see three clusters with distinct size differences. In the dendrogram, we can see there are multiple instances of all three population sizes being placed into the same cluster.

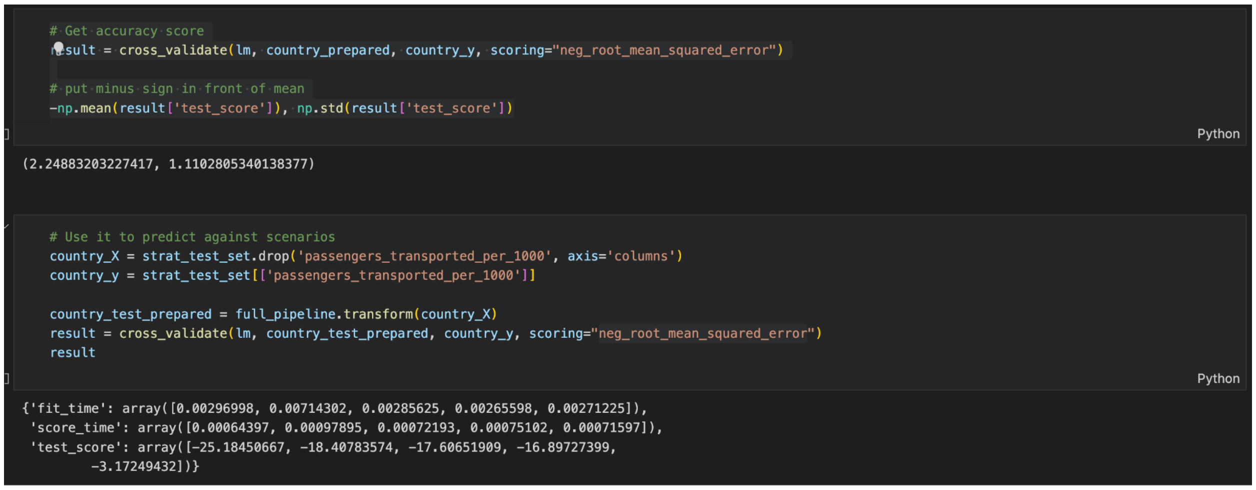
We then wanted to know what these clusters look like compared to the original variable in a scatterplot, so we created pair plots. It isn't possible to determine which cluster corresponds to which population size because there is so much overlap.

```
# make a pairplot that uses the pop_size as the hue
sns.pairplot(data=pop_size_clustering.drop('Y_pred_pop', axis=1), hue='pop_size', aspect=1.5)
plt.gcf().axes[-1].xaxis.set_major_formatter(pop_formatter)
plt.show()

# make a pairplot that uses the clustering results as the hue
sns.pairplot(data=pop_size_clustering, hue='Y_pred_pop', aspect=1.5)
plt.gcf().axes[-1].xaxis.set_major_formatter(pop_formatter)
plt.show()
```

Linear Regression

After performing a clustering analysis on the transportation dataset, it became clear that a simple approach might be most effective. Linear regression is a technique that creates a model to illustrate the relationship between a dependent variable and one or more independent variables. Linear models are interpretable, relatively straightforward to implement, and testable.

We began by considering two Python libraries that create linear regression models: the OLS (Ordinary Least Squares) Regression model from statsmodels, and the LinearRegression model from scikit-learn. While the statsmodels option gives us a host of tools and analytical features so we can interpret the model, scikit-learn is more appropriate to create a predictive model that we can use in the feature. We employed both to compare their performance and insights.



```
# Get accuracy score
result = cross_validate(lm, country_prepared, country_y, scoring="neg_root_mean_squared_error")

# put minus sign in front of mean
-np.mean(result['test_score']), np.std(result['test_score'])
```
Python

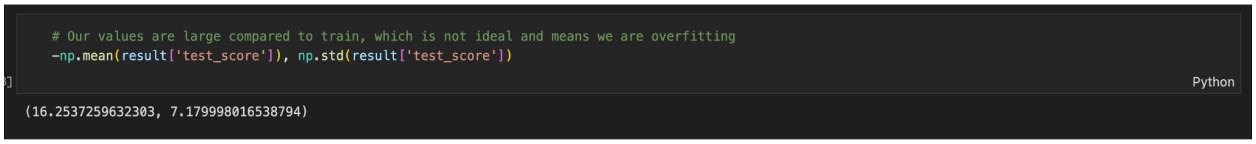
```
(2.24883203227417, 1.1102805340138377)
```

```
# Use it to predict against scenarios
country_X = strat_test_set.drop('passengers_transported_per_1000', axis='columns')
country_y = strat_test_set[['passengers_transported_per_1000']]

country_test_prepared = full_pipeline.transform(country_X)
result = cross_validate(lm, country_test_prepared, country_y, scoring="neg_root_mean_squared_error")
result
```
Python

```
{'fit_time': array([0.00296998, 0.00714302, 0.00285625, 0.00265598, 0.00271225]),
 'score_time': array([0.00064397, 0.00097895, 0.00072193, 0.00075102, 0.00071597]),
 'test_score': array([-25.18450667, -18.40783574, -17.60651909, -16.89727399,
        -3.17249432])}
```

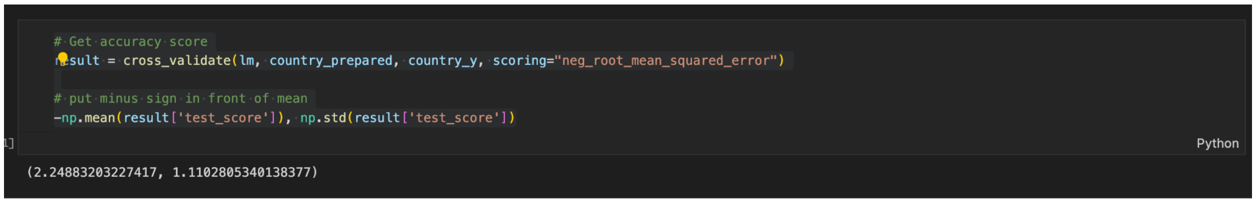
When using the LinearRegression model in scikit-learn, we used a OneHotEncoder to encode categorical attributes in order to be used in the model. After training, fitting, and testing the model, we cross-validated our model and used the negative root mean squared error (RMSE) as a performance metric to show the average squared difference between the predicted and actual values of the independent variable.

The default value for the test size is 0.1. We created two separate models, one with the default setting, and one with 0.33 to see if the proportion of the total dataset being used has any effect on our results. The mean and standard deviation test scores we received for the training data were 2.25 and 1.11, respectively, and 16.25 and 7.18 for the testing data.



```
# Our values are large compared to train, which is not ideal and means we are overfitting
-np.mean(result['test_score']), np.std(result['test_score'])
```
Python

```
(16.2537259632303, 7.179998016538794)
```

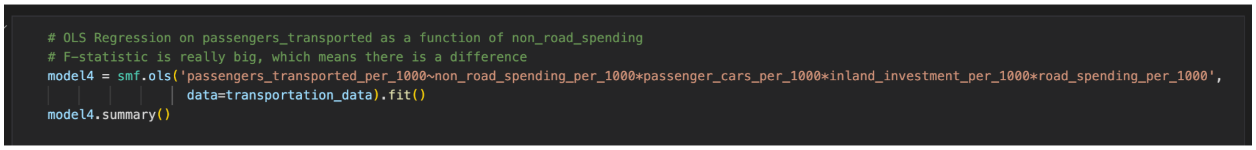


```python
# Get accuracy score
result = cross_validate(lm, country_prepared, country_y, scoring="neg_root_mean_squared_error")

# put minus sign in front of mean
-np.mean(result['test_score']), np.std(result['test_score'])
```

```
(2.24883203227417, 1.1102805340138377)
```

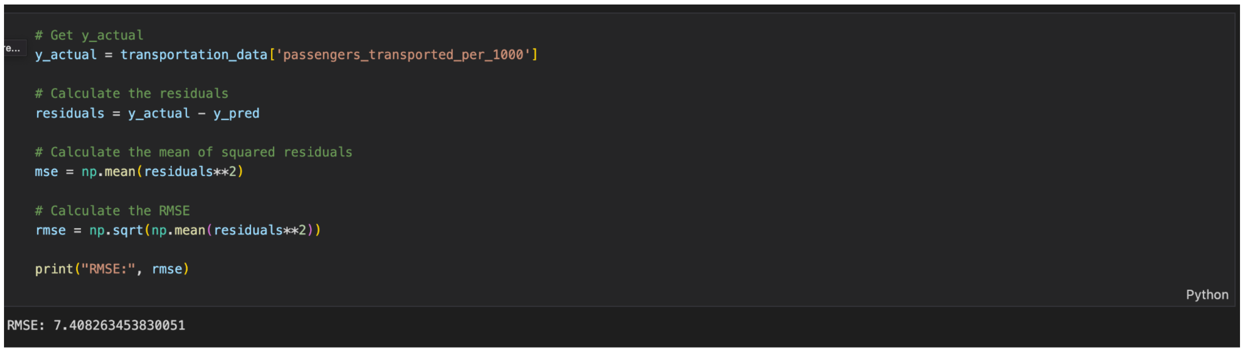
We can see that our accuracy score of the train vs test data differs significantly. The ideal value for an RMSE is close to 0, but our values for the test data are at 16 and 7, which is larger than the values from the train data, meaning that our model is not only prone to errors but is also likely overfitting. But what happens when we create a similar model and increase the test size?

Adjusting the test size to 0.33 made the average and standard deviation of the test scores slightly smaller and larger, respectively. Regardless, these values are still bigger than the train data's results, which tells us that updating the test size did change the results but not in a way that indicates to us that this is a particularly good model to use. Even after some adjustments, we have determined that although creating a linear regression in this format is possible, it might not be ideal. Our dataset is too small given what we are trying to accomplish, so our resulting model will overfit. We will try to simplify our approach by using the statsmodels OLS model and reducing our number of variables.



```python
# OLS Regression on passengers_transported as a function of non_road_spending
# F-statistic is really big, which means there is a difference
model4 = smf.ols('passengers_transported_per_1000~non_road_spending_per_1000*passenger_cars_per_1000*inland_investment_per_1000*road_spending_per_1000',
                  data=transportation_data).fit()
model4.summary()
```

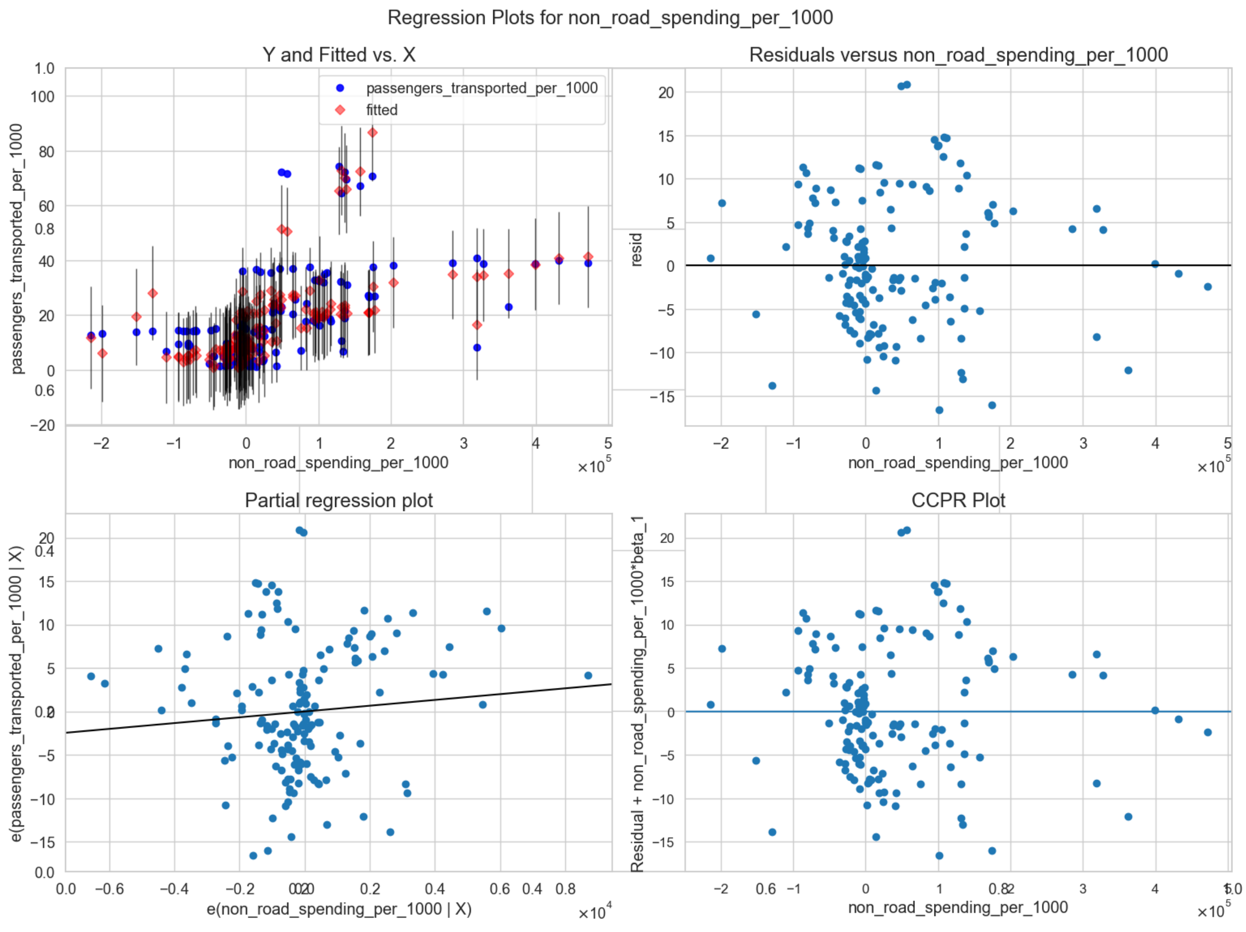
For the OLS model, we added the ‘passengers_transported_per_1000’ as the dependent variable, and the ‘non_road_spending_per_1000’ and ‘passenger_cars_per_1000’ as the independent variables for the first model. We also created a final model that includes those independent variables, and added ‘inland_investment_per_1000’ and ‘road_spending_per_1000' for additional context.



```python
# Get y_actual
y_actual = transportation_data['passengers_transported_per_1000']

# Calculate the residuals
residuals = y_actual - y_pred

# Calculate the mean of squared residuals
mse = np.mean(residuals**2)

# Calculate the RMSE
rmse = np.sqrt(np.mean(residuals**2))

print("RMSE:", rmse)
```

```
RMSE: 7.408263453830051
```

We predicted on both of these models, and calculated the RMSE of both to compare them and see if adding more variables made our results more accurate. Based on our RMSE for these models (13.84 and 7.40, respectively) and their squared values (191.62 and 54.88, respectively) we can conclude that more variables significantly improved our results.



The regression plots above show the behavior of the residuals, with confidence intervals vs. the independent variable chosen, the residuals of the model vs. the chosen independent variable, a partial regression plot, and a CCPR plot. There is a general pattern to be found, but it is still high in variance for the residuals. This means that, although we have managed to decrease our RMSE value and residuals, the model is still not performing at a high degree of reliability.

Conclusion

In sum, we looked at correlations, clustering, and regressions to determine the relationship between investment in transportation and how many people use these services. From our analyses detailed above, we can find that, while there is a general relationship, we did not create models that are highly accurate.

Our major roadblock was the limitation in scope of the data. Because we were pulling data from multiple sources, we had to remove rows with incomplete data. This made our dataset relatively small. When you attempt to model based on smaller datasets, there is a risk of overfitting, underfitting, or not capturing the context needed to create a model that can predict values with an acceptable RMSE. In the future, we would want to conduct similar tests, because regressions are appropriate to predict an outcome based on a few variables, but we would want to make sure we are collecting data on a larger number of countries and features.

Statement of Work

Team Contributions

Each team member was responsible for owning specific portions of the project. Eman pulled the data from the EuroStat and OECD websites and started our initial Jupyter notebook and data cleansing. She also performed the regression analysis. Elaine also contributed to the data cleansing, joining the data, and completing the correlation analysis. Tyler's main responsibility was performing the clustering analysis and assisting in other organizational tasks to create the final product.

As a team, we worked together to create the final Jupyter notebook with our code and markdown comments. We also worked collaboratively on the written portion of the report where each team member wrote the portion for each of the analyses they were responsible for (e.g., Tyler wrote the clustering analysis section). Additionally, Eman took ownership of writing the Motivation and Data Sources section, and rallying our team together.

Future Collaboration Improvements

As we worked to put together the final Jupyter notebook, we discovered that GitHub does not work well with Jupyter notebooks. We started with separate branches, but had issues with merging changes together. In the process of pushing new changes, we ended up with a corrupted notebook that we were unable to use. Thankfully, Elaine had the most recent version saved to her local computer and we were able to recover our work without too many headaches. In the future, when working with teams on Jupyter notebooks, separate branches need to be used when creating and pushing changes. Pushing multiple changes to GitHub to the same branch with the same notebook name caused issues for our team and required additional time to recover.

In addition, in the future, we need to be more careful about our initial data sources and data cleaning. We want to make sure our data is robust enough to perform the chosen analyses on – otherwise our results may be inconclusive or not convincing enough.