An In-depth Examination of parameters contributing to the success of Electric Vehicle business Models

Oluwatoyin Oniroko

Ishmam Zahin Chowdhury

Oklahoma State University

Project Presentation Outline

Context

Recent Status of EV

Bulletins

- Oklahoma received $66 million from the National Electric Vehicle Formula Program (NEVI) to develop charging infrastructure.
- Electric vehicles are piling up at dealership lots
- EV sales took twice as long in August 2023 compared to January of the same year, while gas-powered vehicles continued to sell swiftly.

Discovery

Literature Review

Challenges faced by the EV market

- Struggle to meet government and manufacturer expansion visions.
- Facing competition from established internal combustion engine (ICE) vehicles.
- Slow progress in EV promotion and lack of innovative, EV-tailored business models.
- Traditional ICE-oriented business models aren't aligned with EV-specific characteristics like limited range and higher costs.
- Successful EV commercialization demands unique business models tailored to EV requirements.

Discovery

Project Scope

Scope

Effects of infrastructural development to promote EV sales

Data Preparation

Data Collection

Data Sources

- National Renewable Energy Laboratory (NREL)
- Alternative Fuels Data Center (AFDC)
- World Population Review
- United States Census Bureau
- Bureau of Transportation Statistics
- Statista
- Kaggle
- Others

Data Preparation

Cleaning and Preparing

Libraries

Pandas

US

Data Sheets to prepare

Charging stations

EV registration

Household/ household income/ travel tendency

Cost of Ownership

Master data sheet

Model Planning

Data Analysis

Independent Variables

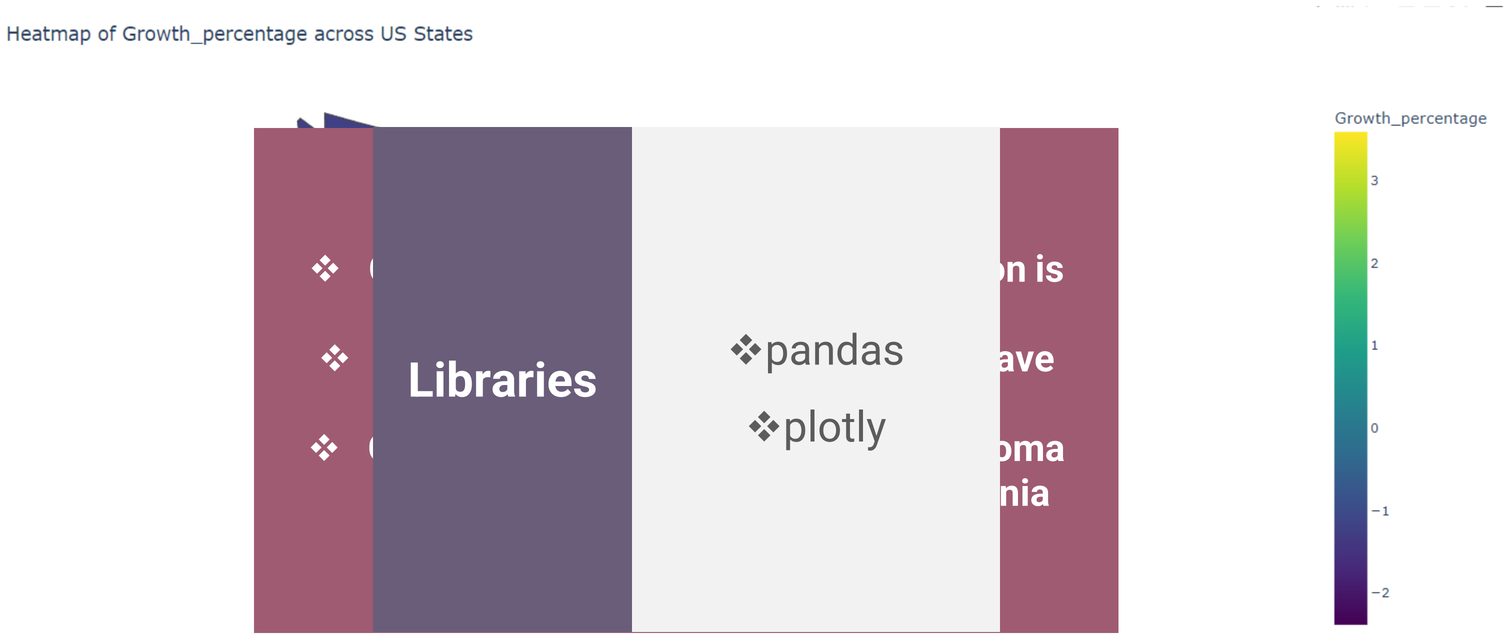
- Average Mileage
- Average trip counts
- Average Mileage per trips
- Total Cost without purchase price of EV
- Total Cost with purchase price of EV
- Taxes
- Fuel
- Insurance
- Non-EV vs EV difference in Total cost without a purchase price
- Non-EV vs EV difference in Total cost with purchase price
- Non-EV vs EV difference in Taxes
- Non-EV vs EV difference in Fuel
- Non-EV vs EV difference in Insurance
- Median Income
- Total Count
- Percentage
- EV Level1 EVSE Percentage
- EV Level2 EVSE Percentage
- EV DC Fast Count Percentage

Dependent Variables

- Count of EV Registration
- Probability of Having EVs per Household
- Growth Percentage

Model Planning

Descriptive Analysis



Libraries

- pandas
- plotly

Model Planning

Descriptive and Inferential Analysis

Feature selection

Most contributing Features-(information gain)

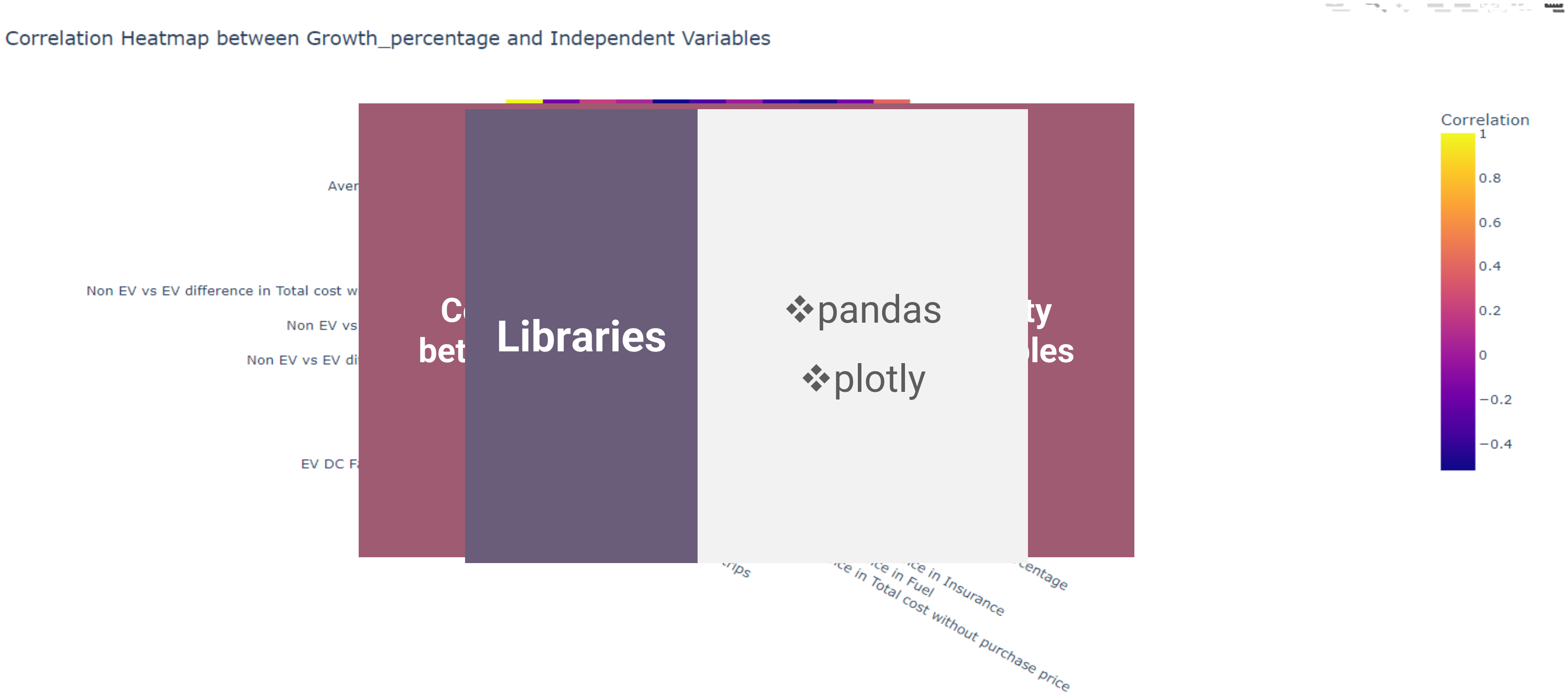
- **Median Income**
- **Non-EV vs EV difference in Total cost without purchase price**
- **Total Count**
- **Fuel**
- **Taxes**

Libraries

- pandas
- sklearn

Model Planning

Descriptive and Inferential Analysis



Libraries

- ❖pandas
- ❖plotly

Model Planning

Descriptive and Inferential Analysis

Variance Inflation Factor-

To remove multicollinearity

Selected independent variables-

- Average trip counts
- Average Mileage per trips
- Taxes
- Fuel
- Non-EV vs EV difference in Total cost without a purchase price
- Non-EV vs EV difference in Fuel
- Non-EV vs EV difference in Insurance
- Median Income
- Percentage
- EV DC Fast Count Percentage

Libraries

- pandas
- statmodels

Result and Discussion

Predictive Modeling

Model	Dependent Variable			Significance	R^2
Backward Eliminating MLR	Growth Percentage			0.001	0.609
				0.004	
				0.008	
				0.017	
		EV DC fast count Percentage	0.4285	0.001	
		Percentage	0.4285	0.000	

Libraries

- pandas
- statmodels

Communicating Results

Charging infrastructure	Mileage and trip counts	Cost of Ownership vs Median income
Frequent charging infrastructure with better fast charge ability	Infrastructure setup should be in consideration with urban distances	Market segmentation based on median income and corresponding cost of ownerships

Limitations of project

Direction for Future Studies/ operationalize

Limitation1
01
Incorporating customer feedback data

Limitation 2
02
Incorporating chronological set of data

Limitation 3
03
incorporating aspects of EV production, battery capacity, and other facilities

References

- ❖ Breetz, H. L., & Salon, D. (2018). Do electric vehicles need subsidies? Ownership costs for conventional, hybrid, and electric vehicles in 14 US cities. *Energy Policy*, *120*, 238–249.
- ❖ Conejero, J. A., Jordán, C., & Sanabria-Codesal, E. (2014). An iterative algorithm for the management of an electric car-rental service. *Journal of Applied Mathematics*, *2014*.
- ❖ Danielis, R., Rotaris, L., Giansoldati, M., & Scorrano, M. (2020). Drivers' preferences for electric cars in Italy. Evidence from a country with limited but growing electric car uptake. *Transportation Research Part A: Policy and Practice*, *137*, 79–94.
- ❖ Dijk, M., Wells, P., & Kemp, R. (2016). Will the momentum of the electric car last? Testing an hypothesis on disruptive innovation. *Technological Forecasting and Social Change*, *105*, 77–88.
- ❖ Dorcec, L., Pevec, D., Vdovic, H., Babic, J., & Podobnik, V. (2019). How do people value electric vehicle charging service? A gamified survey approach. *Journal of Cleaner Production*, *210*, 887–897.
- ❖ Secinaro, S., Brescia, V., Calandra, D., & Biancone, P. (2020). Employing bibliometric analysis to identify suitable business models for electric cars. *Journal of Cleaner Production*, *264*, 121503.
- ❖ Secinaro, S., Calandra, D., Lanzalonga, F., & Ferraris, A. (2022). Electric vehicles' consumer behaviours: Mapping the field and providing a research agenda. *Journal of Business Research*, *150*, 399–416.
- ❖ Umar, M., Ji, X., Kirikkaleli, D., & Alola, A. A. (2021). The imperativeness of environmental quality in the United States transportation sector amidst biomass-fossil energy consumption and growth. *Journal of Cleaner Production*, *285*, 124863.
- ❖ Yang, Y., & Tan, Z. (2019). Investigating the influence of consumer behavior and governmental policy on the diffusion of electric vehicles in Beijing, China. *Sustainability*, *11*(24), 6967.
- ❖ Zhang, L., Zhao, Z., Xin, H., Chai, J., & Wang, G. (2018). Charge pricing model for electric vehicle charging infrastructure public-private partnership projects in China: A system dynamics analysis. *Journal of Cleaner Production*, *199*, 321–333.
- ❖ Ziegler, D., & Abdelkafi, N. (2022). Business models for electric vehicles: Literature review and key insights. *Journal of Cleaner Production*, *330*, 129803.

THANK YOU !

An In-depth Examination of parameters contributing to the success of Electric Vehicle business Models

Oluwatoyin Oniroko

Ishmam Zahin Chowdhury

Oklahoma State University

QUESTIONS

Supplementary slides

Coding for Data Visuals

```
import plotly.express as px
import pandas as pd

# Load your dataset here (replace 'your_dataset.csv' with your dataset file)
# For this example, using Plotly's built-in dataset
df = pd.read_csv("C:\\Users\ishma\Desktop\Courseworks\msisprogramming\Project work\df43.csv")

# Define the variables
dependent_variable_1= 'Count of EV registration (2022)'
dependent_variable_2= 'Growth_percentage'
dependent_variable_3= 'Probability of having EVs per household'
independent_variables = ['Average trip counts', 'Average Mileage per trips',
                    'Taxes', 'Fuel',
                    'Non EV vs EV difference in Total cost without purchase price',
                    'Non EV vs EV difference in Fuel', 'Non EV vs EV difference in Insurance',
                    'Median Income', 'Percentage', 'EV DC Fast Count Percentage']

# Calculate the correlation matrix
correlation_matrix_1 = df[[dependent_variable_1] + independent_variables].corr()
correlation_matrix_2 = df[[dependent_variable_2] + independent_variables].corr()
correlation_matrix_3 = df[[dependent_variable_3] + independent_variables].corr()
# Create the heatmap
fig_1 = px.imshow(correlation_matrix_1,
           labels=dict(color='Correlation'),
           x=correlation_matrix_1.columns,
           y=correlation_matrix_1.columns,
           title=f'Correlation Heatmap between {dependent_variable_1} and Independent Variables')

fig_2 = px.imshow(correlation_matrix_2,
           labels=dict(color='Correlation'),
           x=correlation_matrix_2.columns,
           y=correlation_matrix_2.columns,
           title=f'Correlation Heatmap between {dependent_variable_2} and Independent Variables')

fig_3 = px.imshow(correlation_matrix_3,
           labels=dict(color='Correlation'),
           x=correlation_matrix_3.columns,
           y=correlation_matrix_3.columns,
           title=f'Correlation Heatmap between {dependent_variable_3} and Independent Variables')

# Show the heatmap
fig_1.show()
fig_2.show()
fig_3.show()
```

```
import plotly.express as px
import pandas as pd

# Load your dataset here (replace 'your_dataset.csv' with your dataset file)
# For this example, using Plotly's built-in dataset
df = pd.read_csv("C:\\Users\ishma\Desktop\Courseworks\msisprogramming\Project work\df43.csv")

# Define your dependent variable and independent variables
dependent_variable_1 = 'Count of EV registration (2022)'
dependent_variable_2 = 'Growth_percentage'
dependent_variable_3= 'Probability of having EVs per household'
independent_variables = ['Average trip counts', 'Average Mileage per trips',
               'Taxes', 'Fuel',
               'Non EV vs EV difference in Total cost without purchase price',
               'Non EV vs EV difference in Fuel', 'Non EV vs EV difference in Insurance',
               'Median Income', 'Percentage', 'EV DC Fast Count Percentage'] # Add more independent variables

# Create a heatmap based on US states
fig_1 = px.choropleth(df, # Pass the DataFrame directly
               locations='State', # Assuming 'State' contains state codes or names
               locationmode='USA-states',
               color=dependent_variable_1,
               scope='usa',
               labels={'color': dependent_variable_1},
               title=f'Heatmap of {dependent_variable_1} across US States',
               color_continuous_scale='Viridis') # Change the color scale if needed

fig_2 = px.choropleth(df, # Pass the DataFrame directly
               locations='State', # Assuming 'State' contains state codes or names
               locationmode='USA-states',
               color=dependent_variable_2,
               scope='usa',
               labels={'color': dependent_variable_2},
               title=f'Heatmap of {dependent_variable_2} across US States',
               color_continuous_scale='Viridis') # Change the color scale if needed

fig_3= px.choropleth(df, # Pass the DataFrame directly
               locations='State', # Assuming 'State' contains state codes or names
               locationmode='USA-states',
               color=dependent_variable_3,
               scope='usa',
               labels={'color': dependent_variable_3},
               title=f'Heatmap of {dependent_variable_3} across US States',
               color_continuous_scale='Viridis') # Change the color scale if needed

# Show the heatmap
fig_1.show()
fig_2.show()
fig_3.show()
```

Supplementary slides

Coding for Feature Selection and VIF

```
### Feature Selection
import pandas as pd
import sklearn.feature_selection as fs

df = pd.read_csv ("C:\\Users\ishma\Desktop\Courseworks\msisprogramming\Project work\Electric_vehicle_statewise_parametric_attributes_2.csv")
print(df.head())

# Specify the columns to be considered as independent variables
selected_columns = ['Average Mileage', 'Average trip counts', 'Average Mileage per trips','Total Cost without purchase price of EV',
          'Total Cost with purchase price of EV','Taxes','Fuel','Insurance',
          'Non EV vs EV difference in Total cost without purchase price',
          'Non EV vs EV difference in Total cost with purchase price','Non EV vs EV difference in Taxes',
          'Non EV vs EV difference in Fuel','Non EV vs EV difference in Insurance','Median Income','Total Count',
          'Percentage','EV Level1 EVSE Percentage','EV Level2 EVSE Percentage','EV DC Fast Count Percentage']

print(df[selected_columns])
print(df["Count of EV registration (2022)"])
## for df
X1 = df[selected_columns]
y11 = df['Count of EV registration (2022)']
y12= df['Growth_percentage']
y13= df['Probability of having EVs per household']

# Calculate Information Gain for each feature
information_gain_11= dict(zip(selected_columns,fs.mutual_info_regression(X1, y11, discrete_features=False)))
information_gain_12= dict(zip(selected_columns,fs.mutual_info_regression(X1, y12, discrete_features=False)))
information_gain_13= dict(zip(selected_columns,fs.mutual_info_regression(X1, y13, discrete_features=False)))

print(information_gain_11)
print(information_gain_12)
print(information_gain_13)


print("=========1")
# Function to sort and print the information gain scores
def print_sorted_information_gain(information_gain):
  sorted_info_gain = sorted(information_gain.items(), key=lambda x: x[1], reverse=True)
  for feature, score in sorted_info_gain:
    print(f"{feature}: {score}")

# Print the sorted information gain scores
print("Information Gain for y11 (Count of EV registration (2022)):")
print_sorted_information_gain(information_gain_11)
print("\nInformation Gain for y12 (Growth_percentage):")
print_sorted_information_gain(information_gain_12)
print("\nInformation Gain for y13 (Probability of having EVs per household):")
print_sorted_information_gain(information_gain_13)

print('======')
# Function to get the top N features
def top_n_features(information_gain, n=15):
  sorted_info_gain = sorted(information_gain.items(), key=lambda x: x[1], reverse=True)
  return dict(sorted_info_gain[:n])

# Get top 10 most contributing features common for y11, y12, y13, and y14
top_features_y11 = set(top_n_features(information_gain_11).keys())
top_features_y12 = set(top_n_features(information_gain_12).keys())
top_features_y13 = set(top_n_features(information_gain_13).keys())

common_top_features = list(top_features_y11 & top_features_y12 & top_features_y13)
print("Top 12 most contributing features common for y11, y12, and y13:")
print(common_top_features)
```

```
import pandas as pd
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

# Loading the dataset
df = pd.read_csv(
  "C:\\Users\ishma\Desktop\Courseworks\msisprogramming\Project work\Electric_vehicle_statewise_parametric_attributes_2.csv")
print(df.head())

# Select only the columns used as independent variables
independent_cols = ['Average Mileage', 'Average trip counts', 'Average Mileage per trips',
          'Total Cost without purchase price of EV',
          'Total Cost with purchase price of EV', 'Taxes', 'Fuel', 'Insurance',
          'Non EV vs EV difference in Total cost without purchase price',
          'Non EV vs EV difference in Total cost with purchase price', 'Non EV vs EV difference in Taxes',
          'Non EV vs EV difference in Fuel', 'Non EV vs EV difference in Insurance', 'Median Income',
          'Total Count',
          'Percentage', 'EV Level1 EVSE Percentage', 'EV Level2 EVSE Percentage',
          'EV DC Fast Count Percentage']
X = df[independent_cols]

# Iteratively drop features with the highest VIF until reaching 10 least VIF features
while len(independent_cols) > 10:
  # Add a constant to the independent variables matrix (required for VIF calculation)
  X_with_const = add_constant(X)

  # Calculate VIF for each variable, excluding 'const'
  vif_data = pd.DataFrame()
  vif_data["Features"] = X_with_const.columns
  vif_data["VIF"] = [variance_inflation_factor(X_with_const.values, i) for i in range(X_with_const.shape[1])]

  # Remove 'const' from the list of features
  vif_data = vif_data[vif_data["Features"] != "const"]

  # Get the feature with the highest VIF value
  max_vif_feature = vif_data.loc[vif_data['VIF'].idxmax()]['Features']

  # Drop the feature with the highest VIF value
  X = X.drop(columns=max_vif_feature)
  independent_cols.remove(max_vif_feature)
  print(f"Dropped: {max_vif_feature}")

# Print the 10 least VIF features
print(f"\nRemaining Features with 10 Least VIF Values:")
print(vif_data.nsmallest(10, 'VIF'))
```

Supplementary slides

Coding for the predictive models

```
### Feature Selection
import pandas as pd
import sklearn.feature_selection as fs

df = pd.read_csv ("C:\\Users\ishma\Desktop\Courseworks\msisprogramming\Project work\Electric_vehicle_statewise_parametric_attributes_2.csv")
print(df.head())

# Specify the columns to be considered as independent variables
selected_columns = ['Average Mileage', 'Average trip counts', 'Average Mileage per trips','Total Cost without purchase price of EV',
            'Total Cost with purchase price of EV','Taxes','Fuel','Insurance',
            'Non EV vs EV difference in Total cost without purchase price',
            'Non EV vs EV difference in Total cost with purchase price','Non EV vs EV difference in Taxes',
            'Non EV vs EV difference in Fuel','Non EV vs EV difference in Insurance','Median Income','Total Count',
            'Percentage','EV Level1 EVSE Percentage','EV Level2 EVSE Percentage','EV DC Fast Count Percentage']

print(df[selected_columns])
print(df["Count of EV registration (2022)"])
## for df
X1 = df[selected_columns]
y11 = df['Count of EV registration (2022)']
y12= df['Growth_percentage']
y13= df['Probability of having EVs per household']

# Calculate Information Gain for each feature
information_gain_11= dict(zip(selected_columns,fs.mutual_info_regression(X1, y11, discrete_features=False)))
information_gain_12= dict(zip(selected_columns,fs.mutual_info_regression(X1, y12, discrete_features=False)))
information_gain_13= dict(zip(selected_columns,fs.mutual_info_regression(X1, y13, discrete_features=False)))

print(information_gain_11)
print(information_gain_12)
print(information_gain_13)


print("=========1")
# Function to sort and print the information gain scores
def print_sorted_information_gain(information_gain):
   sorted_info_gain = sorted(information_gain.items(), key=lambda x: x[1], reverse=True)
   for feature, score in sorted_info_gain:
      print(f"{feature}: {score}")

# Print the sorted information gain scores
print("Information Gain for y11 (Count of EV registration (2022)):")
print_sorted_information_gain(information_gain_11)
print("\nInformation Gain for y12 (Growth_percentage):")
print_sorted_information_gain(information_gain_12)
print("\nInformation Gain for y13 (Probability of having EVs per household):")
print_sorted_information_gain(information_gain_13)

print('======')
# Function to get the top N features
def top_n_features(information_gain, n=15):
   sorted_info_gain = sorted(information_gain.items(), key=lambda x: x[1], reverse=True)
   return dict(sorted_info_gain[:n])

# Get top 10 most contributing features common for y11, y12, y13, and y14
top_features_y11 = set(top_n_features(information_gain_11).keys())
top_features_y12 = set(top_n_features(information_gain_12).keys())
top_features_y13 = set(top_n_features(information_gain_13).keys())

common_top_features = list(top_features_y11 & top_features_y12 & top_features_y13)
print("Top 12 most contributing features common for y11, y12, and y13:")
print(common_top_features)
```