Comprehensive Data Analysis Report

Report generated on 2025-04-27 20:25:41

Final Dataset Snapshot

Shape: (235692, 18)

Columns: VIN (1-10), County, City, State, Postal Code, Model Year, Make, Model, Electric Vehicle Type, Clean Alternative Fuel Vehicle (CAFV) Eligibility, Electric Range, Base MSRP, Legislative District, DOL Vehicle ID, Vehicle Location, Electric Utility, 2020 Census Tract, Vehicle Age

Sample Rows:

VIN (1-10) County City State Postal Code Model Year Make Model Electric Vehicle Type Clean Alternative Fuel Vehicle (CAFV) Eligibility Electric Range Base MSRP Legislative District DOL Vehicle ID Vehicle Location Electric Utility 2020 Census Tract Vehicle Age

KNDAEFS53R King Kent WA 98032 2024 KIA EV9 Battery Electric Vehicle (BEV) Eligibility unknown as battery range has not been researched 0.0 0.0 33.0 278438716 POINT (-122.23741 47.3807) PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA) 5.303303e+10 1

7SAYGAEE6P King Shoreline WA 98155 2023 TESLA MODEL Y Battery Electric Vehicle (BEV) Eligibility unknown as battery range has not been researched 0.0 0.0 32.0 262245623 POINT (-122.3175 47.75781) CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA) 5.303302e+10 2

KNDCE3LG6K Snohomish Lake Stevens WA 98258 2019 KIA NIRO Battery Electric Vehicle (BEV) Clean Alternative Fuel Vehicle Eligible 239.0 0.0 39.0 125809911 POINT (-122.06402 48.01497) PUGET SOUND ENERGY INC 5.306105e+10 6

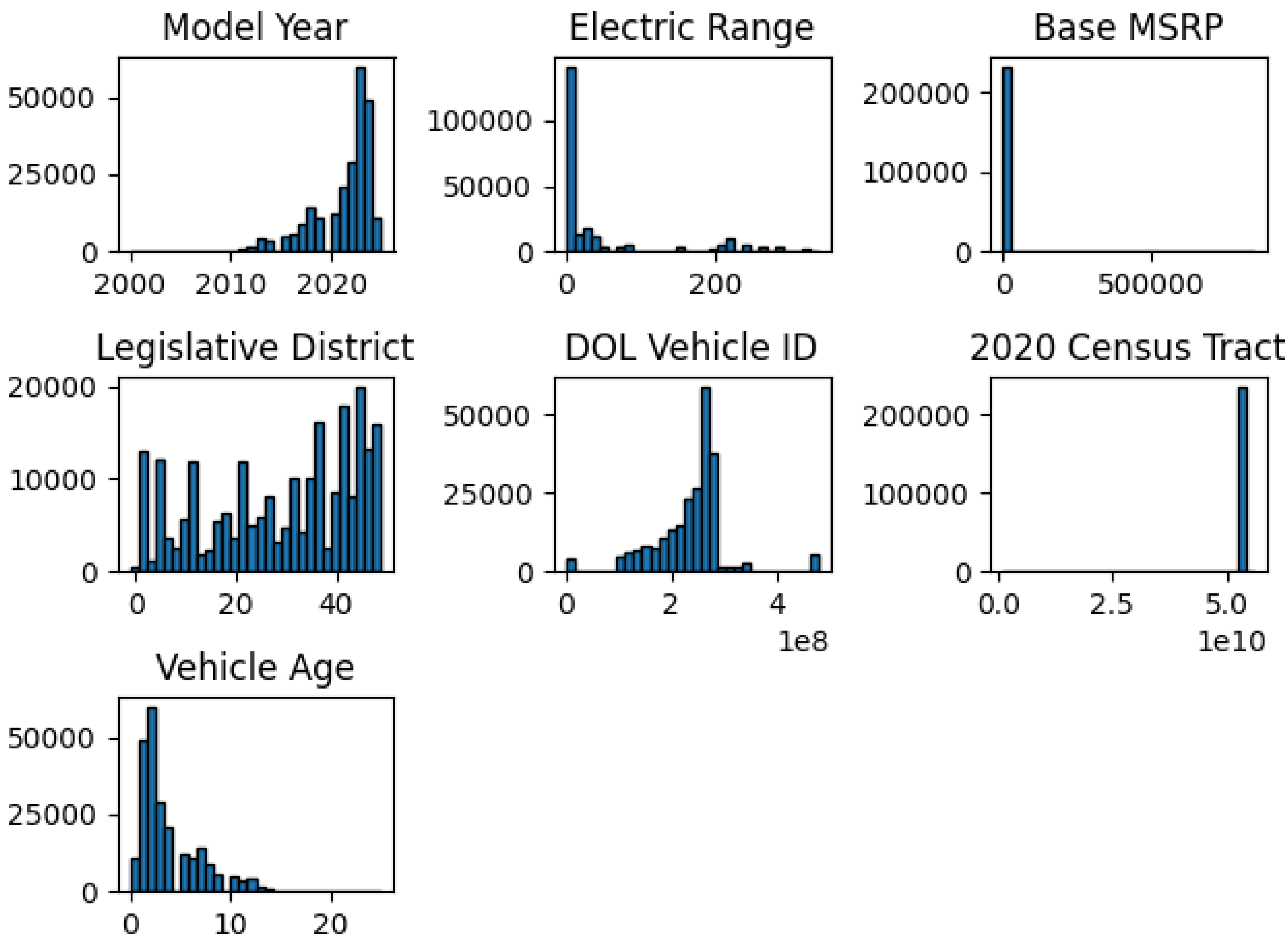
5YJ3E1EA8J Kitsap Bainbridge Island WA 98110 2018 TESLA MODEL 3 Battery Electric Vehicle (BEV) Clean Alternative Fuel Vehicle Eligible 215.0 0.0 23.0 270523354 POINT (-122.521 47.62728) PUGET SOUND ENERGY INC 5.303509e+10 7

7SAYGDEF2N Snohomish Everett WA 98201 2022 TESLA MODEL Y Battery Electric Vehicle (BEV) Eligibility unknown as battery range has not been researched 0.0 0.0 38.0 207408309 POINT (-122.20596 47.97659) PUGET SOUND ENERGY INC

Comprehensive Data Analysis Report

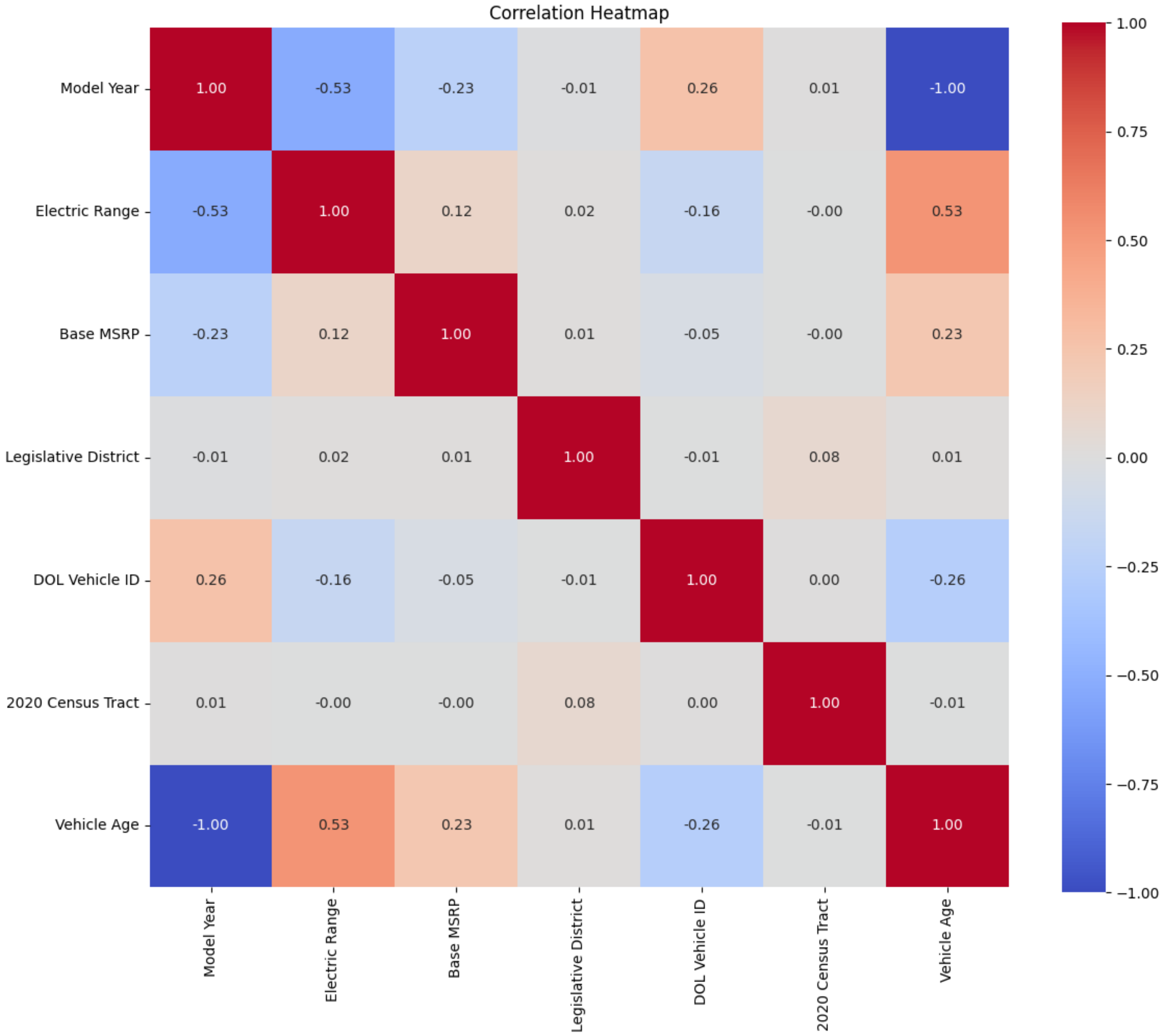
5.306104e+10 3

Data Distributions



Correlation Heatmap

Comprehensive Data Analysis Report



Analyzer Output

The Analyzer agent performs an initial check on the dataset to ensure it is structurally valid and highlights any potential issues. Here are the results:

technical_profile: {'shape': (235692, 17), 'columns': ['VIN (1-10)', 'County', 'City', 'State', 'Postal Code', 'Model Year', 'Make', 'Model', 'Electric Vehicle Type', 'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Electric Range', 'Base MSRP', 'Legislative District', 'DOL Vehicle ID', 'Vehicle Location', 'Electric Utility', '2020 Census Tract'], 'dtypes': {'VIN (1-10)': 'object', 'County': 'object', 'City': 'object', 'State': 'object', 'Postal Code': 'float64', 'Model Year': 'int64', 'Make': 'object', 'Model': 'object', 'Electric Vehicle Type': 'object', 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 'object', 'Electric Range': 'float64', 'Base MSRP': 'float64', 'Legislative District': 'float64', 'DOL Vehicle ID': 'int64', 'Vehicle Location': 'object', 'Electric Utility': 'object', '2020 Census Tract':

'float64'}, 'missing_values': {'VIN (1-10)': 0, 'County': 3, 'City': 3, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3}, 'memory_usage': '169800.34 KB'}

ai_analysis: {'structure': 'Section not found', 'quality': 'Section not found', 'context': 'Section not found', 'recommendations': ["df['County'].fillna(df['County'].mode()[0], inplace=True)", "df['City'].fillna(df['City'].mode()[0], inplace=True)", "df['Electric Utility'].fillna(df['Electric Utility'].mode()[0], inplace=True)", "df['Postal Code'].fillna(df['Postal Code'].median(), inplace=True)", "df['Electric Range'].fillna(df['Electric Range'].median(), inplace=True)", "df['Base MSRP'].fillna(df['Base MSRP'].median(), inplace=True)", "df['2020 Census Tract'].fillna(df['2020 Census Tract'].median(), inplace=True)", "df['Legislative District'].fillna(-1, inplace=True)", "df['Vehicle Location'].fillna('Unknown', inplace=True)", "df = df[df['Electric Range'] <= 1000]", "df = df[df['Base MSRP'] <= 200000]", "df = df[df['Model Year'] >= 2000]", "df['Postal Code'] = df['Postal Code'].astype(int)", "df['Legislative District'] = df['Legislative District'].astype(int)", "location_string = location_string.replace('POINT (','')", "location_string = location_string.replace(')','')", "df['Latitude'], df['Longitude'] = zip(*df['Vehicle Location'].apply(extract_coordinates))", "df['Latitude'].fillna(df['Latitude'].median(), inplace=True)", "df['Longitude'].fillna(df['Longitude'].median(), inplace=True)", "df['Model Year'] = pd.to_datetime(df['Model Year'], format='%Y')"]}

preprocessing_ready: {'missing': ["df['County'].fillna(df['County'].mode()[0], inplace=True)", "df['City'].fillna(df['City'].mode()[0], inplace=True)", "df['Electric Utility'].fillna(df['Electric Utility'].mode()[0], inplace=True)", "df['Postal Code'].fillna(df['Postal Code'].median(), inplace=True)", "df['Electric Range'].fillna(df['Electric Range'].median(), inplace=True)", "df['Base MSRP'].fillna(df['Base MSRP'].median(), inplace=True)", "df['2020 Census Tract'].fillna(df['2020 Census Tract'].median(), inplace=True)", "df['Legislative District'].fillna(-1, inplace=True)", "df['Vehicle Location'].fillna('Unknown', inplace=True)", "df['Latitude'], df['Longitude'] = zip(*df['Vehicle Location'].apply(extract_coordinates))", "df['Latitude'].fillna(df['Latitude'].median(), inplace=True)", "df['Longitude'].fillna(df['Longitude'].median(), inplace=True)"], 'outliers': [], 'transformations': ["df = df[df['Electric Range'] <= 1000]", "df = df[df['Base MSRP'] <= 200000]", "df = df[df['Model Year'] >= 2000]", "df['Postal Code'] = df['Postal Code'].astype(int)", "df['Legislative District'] = df['Legislative District'].astype(int)", "location_string = location_string.replace('POINT (','')", "location_string = location_string.replace(')','')", "df['Model Year'] = pd.to_datetime(df['Model Year'], format='%Y')"]}

Comprehensive Data Analysis Report

Operator Output

The Operator agent performs preprocessing tasks on the dataset. This includes handling missing values, outliers, and other necessary transformations. Here are the operations performed:

executed_operations: [{'operation': "df['County'].fillna(df['County'].mode()[0], inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 3, 'City': 3, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 3, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3}"}, {'operation': "df['City'].fillna(df['City'].mode()[0], inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 3, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3}"}, {'operation': "df['Electric Utility'].fillna(df['Electric Utility'].mode()[0], inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 3, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3}"}, {'operation': "df['Postal Code'].fillna(df['Postal Code'].median(), inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 3, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel

Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3}"}, {'operation': "df['Electric Range'].fillna(df['Electric Range'].median(), inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 36, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3}"}, {'operation': "df['Base MSRP'].fillna(df['Base MSRP'].median(), inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 36, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3}"}, {'operation': "df['2020 Census Tract'].fillna(df['2020 Census Tract'].median(), inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 3} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 0}"}, {'operation': "df['Legislative District'].fillna(-1, inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 494, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 0} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 0, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 0}"}, {'operation': "df['Vehicle Location'].fillna('Unknown', inplace=True)", 'impact': "Missing values {'VIN (1-10)': 0, 'County': 0, 'City': 0,

Comprehensive Data Analysis Report

'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 0, 'DOL Vehicle ID': 0, 'Vehicle Location': 10, 'Electric Utility': 0, '2020 Census Tract': 0} ? {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 0, 'DOL Vehicle ID': 0, 'Vehicle Location': 0, 'Electric Utility': 0, '2020 Census Tract': 0}"}, {'operation': "df['Postal Code'] = df['Postal Code'].fillna(0).astype(int).astype(str)\n\n# Purpose: Standardize State abbreviations. Some might be lowercase, or have extra spaces.\ndf['State'] = df['State'].str.strip().str.upper()", 'impact': "Executed: Purpose: Convert 'Postal Code' to a string and handle missing values. Missing values in numeric columns cause issues."}, {'operation': "import datetime\ncurrent_year = datetime.datetime.now().year\ndf['Vehicle Age'] = current_year - df['Model Year']\n\n# Purpose: Extract Latitude and Longitude from 'Vehicle Location'. Handle missing 'Vehicle Location'\ndf[['Longitude', 'Latitude']] = df['Vehicle Location'].str.extract(r'POINT \\((.*?)\\)').str.split(' ', expand=True)\ndf['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce') #errors='coerce' makes null values.\ndf['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')", 'error': "'DataFrame' object has no attribute 'str'"}]

suggested_operations: [{'purpose': "Purpose: Convert 'Postal Code' to a string and handle missing values. Missing values in numeric columns cause issues.", 'code': "df['Postal Code'] = df['Postal Code'].fillna(0).astype(int).astype(str)\n\n# Purpose: Standardize State abbreviations. Some might be lowercase, or have extra spaces.\ndf['State'] = df['State'].str.strip().str.upper()", 'safe_to_execute': True}, {'purpose': "Purpose: Create a new feature 'Vehicle Age' from 'Model Year' relative to the current year.", 'code': "import datetime\ncurrent_year = datetime.datetime.now().year\ndf['Vehicle Age'] = current_year - df['Model Year']\n\n# Purpose: Extract Latitude and Longitude from 'Vehicle Location'. Handle missing 'Vehicle Location'\ndf[['Longitude', 'Latitude']] = df['Vehicle Location'].str.extract(r'POINT \\((.*?)\\)').str.split(' ', expand=True)\ndf['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce') #errors='coerce' makes null values.\ndf['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')", 'safe_to_execute': True}, {'purpose': "Purpose: Handle inconsistent 'Clean Alternative Fuel Vehicle (CAFV) Eligibility' values. Standardize to a limited set of known values.", 'code': "known_eligibilities = ['Clean Alternative Fuel Vehicle Eligible', 'Not eligible due to income', 'Eligibility unknown as battery range has not been researched']\ndf['Clean Alternative Fuel Vehicle (CAFV) Eligibility'] = df['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].apply(lambda x: x if x in known_eligibilities else 'Other')\n\n# Purpose: Fill missing 'Electric Range' values based on 'Electric Vehicle Type'. For BEVs, fill with the median range for that make/model.\n# If not BEV, fill with 0.\ndef impute_range(row):\n   if pd.isna(row['Electric Range']):\n   if row['Electric Vehicle Type'] == 'Battery

Comprehensive Data Analysis Report

Electric Vehicle (BEV)':\n median_range = df[(df['Make'] == row['Make']) & (df['Model'] == row['Model']) & (df['Electric Vehicle Type'] == 'Battery Electric Vehicle (BEV)'])['Electric Range'].median()\n return median_range if not pd.isna(median_range) else 0 # If no median found, default to 0\n else:\n return 0\n else:\n return row['Electric Range']\n\ndf['Electric Range'] = df.apply(impute_range, axis=1)", 'safe_to_execute': False}]

data_snapshot: {'shape': (235692, 18), 'missing_values': {'VIN (1-10)': 0, 'County': 0, 'City': 0, 'State': 0, 'Postal Code': 0, 'Model Year': 0, 'Make': 0, 'Model': 0, 'Electric Vehicle Type': 0, 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 0, 'Electric Range': 0, 'Base MSRP': 0, 'Legislative District': 0, 'DOL Vehicle ID': 0, 'Vehicle Location': 0, 'Electric Utility': 0, '2020 Census Tract': 0, 'Vehicle Age': 0}, 'dtypes': {'VIN (1-10)': 'object', 'County': 'object', 'City': 'object', 'State': 'object', 'Postal Code': 'object', 'Model Year': 'int64', 'Make': 'object', 'Model': 'object', 'Electric Vehicle Type': 'object', 'Clean Alternative Fuel Vehicle (CAFV) Eligibility': 'object', 'Electric Range': 'float64', 'Base MSRP': 'float64', 'Legislative District': 'float64', 'DOL Vehicle ID': 'int64', 'Vehicle Location': 'object', 'Electric Utility': 'object', '2020 Census Tract': 'float64', 'Vehicle Age': 'int64'}}

processed_df: VIN (1-10) County City ... Electric Utility 2020 Census Tract Vehicle Age

0 5YJ3E1EBXK King Seattle ... CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA) 5.303301e+10 6

1 5YJYGDEE3L Kitsap Poulsbo ... PUGET SOUND ENERGY INC 5.303509e+10 5

2 KM8KRDAF5P Kitsap Olalla ... PUGET SOUND ENERGY INC 5.303509e+10 2

3 5UXTA6C0XM Kitsap Seabeck ... PUGET SOUND ENERGY INC 5.303509e+10 4

4 JTMAB3FV7P Thurston Rainier ... PUGET SOUND ENERGY INC 5.306701e+10 2

... ...

235687 1C4RJXN62R Pierce Tacoma ... BONNEVILLE POWER ADMINISTRATION||CITY OF TACOM... 5.305306e+10 1

235688 5YJSA1E28J Snohomish Stanwood ... BONNEVILLE POWER ADMINISTRATION||PUD 1 OF SNOH... 5.306105e+10 7

235689 3FA6P0SU2F King Redmond ... PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA) 5.303303e+10 10

Comprehensive Data Analysis Report

235690 WA1BCBFZ6P Snohomish Lake Stevens ... PUGET SOUND ENERGY INC 5.306105e+10 2

235691 WBY33AW03P King Issaquah ... PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA) 5.303303e+10 2

[235692 rows x 18 columns]

Scientist Output

The Scientist agent performs an in-depth analysis and modeling based on the cleaned data. It generates insights and suggests recommendations. Here are the findings and insights:

task: regression

target_column: Electric Range

feature_columns: ['Model Year', 'Make', 'Model', 'Vehicle Age']

rationale: Electric Range is a continuous numerical value, making regression the appropriate task. The features Model Year, Make, Model, and Vehicle Age are likely to influence the electric range of a vehicle. Newer models and specific makes/models tend to have different ranges, and the age of the vehicle could affect battery performance.

model_type: RandomForestRegressor

metrics: {'mse': 2097.6400000000012, 'r2': None}

insights: ["Most important feature: 'Model' (0.33)", "Trained to predict 'Electric Range' using 4 features.", 'Okay, let\'s break down the performance data and provide some key trends and recommendations.\n\n**Understanding the Data**\n\n* **MSE (Mean Squared Error):** 2097.64. This indicates the average squared difference between the predicted values and the actual values. A higher MSE suggests poorer model performance. This value, on its own, is hard to interpret without context (e.g., the scale of the target variable). Is the target variable in the hundreds? Thousands? Millions?\n\n* **R2 (R-squared):** NaN (Not a Number). This means that the R-squared value could not be computed. R-squared represents the proportion of variance in the dependent variable that is predictable from the independent variable(s). A value of NaN usually indicates one of the following:\n\n * **Zero Variance in Target:** The target variable has no variance (all values are the same). The model *might* be predicting the constant value, but there\'s nothing to correlate with.\n * **Identical Predictions:** The model is predicting the *same* value for every input, leading to a division by zero during R-squared calculation. Essentially, the model isn\'t learning anything.\n * **Perfect Prediction (Possibly a Bug):** In rare cases, if the model predicts perfectly (but there\'s an issue in the calculation), you might get NaN. This is less likely.\n * **Data Issue:** There might be issues within the dataset that causes calculation error.\n\n**Key Trends**\n\n1. **Poor Model Fit:** The relatively high MSE indicates that the model is not accurately predicting the target variable.\n\n2. **Major Problem Indicated by NaN R2:** The NaN R2 value is a significant red flag. It signals a fundamental problem with the model, the data, or the calculation of the metric. It suggests the model is likely failing to capture the underlying relationships in the data, or that something is preventing the calculation of this important metric.\n\n**Recommendations**\n\nGiven the severity of the performance issues, I\'d recommend the following actions, in roughly this order:\n\n1. **Investigate the Target Variable:**\n * **Check for Constant Values:** The *very first* thing to do is to confirm if the target variable has zero variance. If every value is the same, there\'s nothing for the model to learn.\n * **Examine Distribution:** Look at the distribution of your target variable (histogram, boxplot). Are there outliers or unusual patterns? Are values missing? Are values as expected?\n * **Scale of the Target Variable:** Understand the units and magnitude of the target variable. This provides context for the MSE value.\n\n2. **Examine Model Predictions:**\n * **Check for Constant Predictions:** See if your model is simply predicting the *same* value for all inputs. This would also explain the NaN R2.\n * **Review Loss Function:** If you have outliers, make sure your loss function is robust against them.\n * **Review Training and Validation:** Confirm the MSE in both training and validation sets. It might also be caused by training error that is not decreasing over epochs.\n\n3. **Data Quality Checks:**\n * **Missing Values:** Check for missing values in your input features. Missing data can wreak havoc, especially if not handled properly.\n * **Data Scaling/Normalization:** Ensure your features are properly scaled or normalized. This can improve the training of many models.\n * **Outliers:** Investigate outliers in your features and target. Consider whether they are legitimate data points or errors. Decide whether to remove, transform, or use a model robust to outliers.\n * **Data Type:** Confirm all datatypes are correct. Check for any issues such as numerical data being interpreted as objects.\n * **Feature Engineering:** If your dataset is too small to draw meaningful conclusions, consider collecting more data or performing feature engineering to add more relevant information to your dataset.\n\n4. **Model Issues:**\n * **Model Complexity:** Consider whether your model is too complex (overfitting) or too simple (underfitting) for the data. Try simpler or more regularized models if overfitting. Try more complex models if underfitting.\n * **Hyperparameter Tuning:** Experiment with different hyperparameters for your model. Use techniques like grid search or random search to find optimal values.\n * **Feature Selection:** Perhaps some of your features are irrelevant or noisy.

Comprehensive Data Analysis Report

Try feature selection techniques (e.g., using feature importance from a tree-based model or techniques like SelectKBest).\n * **Algorithm Selection:** Consider whether the chosen model is appropriate for your data and problem type. Perhaps another model family would be more suitable.\n * **Bug in Code:** Double-check the code you\'re using to calculate R-squared and MSE. A simple programming error could be the cause of the NaN value.\n\n**Example Code Snippets (Illustrative)**\n\n```python\nimport numpy as np\nimport pandas as pd\nfrom sklearn.metrics import mean_squared_error, r2_score\n\n#Example Dataframe\ndf = pd.DataFrame({\'actual\': [1,2,3,4,5], \'predicted\': [1,1,1,1,1]})\n\n# Check for constant target variable\nif len(set(df[\'actual\'])) == 1:\n print("Target variable is constant!")\n\n# Calculate R-squared and handle potential errors\ntry:\n r2 = r2_score(df[\'actual\'], df[\'predicted\'])\n print(f"R-squared: {r2}")\nexcept ValueError as e:\n print(f"Error calculating R-squared: {e}")\n```\n\n**In Summary**\n\nThe current performance is not acceptable. The NaN R2 is a critical issue that needs to be addressed first. Thoroughly investigate the data, model predictions, and the calculation of the R-squared to identify the root cause. From there, systematically address the issues you uncover. Good luck!']

model_path: saved_models/RandomForestRegressor_Electric Range_20250427_202533.pkl

Comprehensive Research Summary

Gemini AI summary failed: object str can't be used in 'await' expression