eBay Used Car Sales Price Prediction

Liang Yang
Department of Electrical and Computer Engineering, University of California San Diego
liy027@ucsd.edu, PID: A69030304,

1 Introduction

There is an ongoing demand in the used car trading market as people have increasing needs of traveling. The pricing evaluation of a used car is a complicated topic and does not have a unified pricing algorithm. This is because as a special E-commerce product, used cars trading are more complex than other traditional E-commerce product. The difficulty of used car pricing lies in the fact that every used car has its unique condition, its pricing is not only dependent on the brand, model and other basic configuration of the used car such as its engine, drive type, it also depends on the mileage, used-time duration, it is even related to the geographic location and the year that it is for sold. Therefore, it is essential to construct an accurate pricing evaluation model for used cars. An accurate pricing model will booster the sales of used car, which is beneficial for an online e-commerce platform such as eBay. In this project, we will do data exploration on the eBay used car sales data, and implement models for used car pricing evaluation using Linear Regression, XGBoost and Neural Network.

2 Dataset

The dataset of the project comes from the used car sales records over a period of 20 months in 2019 and 2020 that are scrapped from eBay [1]. Each sample contains information about a used car sale, like selling price, location, details about the car (*Make, Model, year, Mileage, etc*).

2.1 Data Statistics

There are totally 122144 pieces of data. A typical data records is with columns

ID,pricesold,yearsold,zipcode,Mileage,Make, Model,Year,Trim,Engine,BodyType,NumCylinders, DriveType

where ”id” is the unique identifier of the sales record, ”pricesold” is the price that the used car was actually sold, and it is the target value that we need to make predictions on. The feature ”yearsold” should be the year that this used cars was eventually sold, and ”Year” means the year that this car was manufactured. We need to classify numerical features and categorical features. It can be interpreted from their data type and their actual meaning that *”pricesold,yearsold,Mileage,Year,NumCylinders* are numerical features, and *”zipcode,Make,Model,Trim,Engine,BodyType,DriveType”* are categorical features.

2.2 Data Pre-processing

We need to check for duplicate rows, that is because eBay bids on cars are non-binding [2], the same seller may sell the same car again so then we need to remove the previous sales as it was not a successful sale. We check for duplicate by checking whether there are duplicate ”ID”. Fortunately, all ”ID”s are unique. Moreover, since ”ID” does not carry additional meaning apart from indexing, we can drop this column.

We need to check for missing data. By doing data visualization for the the missing values as in **Figure 1** **(Note: All figures are in *section 4 Appendix*)**, we can get a more intuitive view on which feature has most data missing. We find that feature ”Trim” has 40.05% data missing, ”Engine” has 22.19% missing, ”BodyType” has 17.02% missing, ”DriveType” has 20.35% missing, ”zipcode” has 0.744% missing, ”model” has 0.469% missing. Since the missing rate for ”Trim” is too high, and its values are too sparse (not unified, unlike ”DriveType” that most values take ”AWD” or ”4WD”), and it does not carry informative mean-

ing, we decide to drop the column "Trim". Moreover, since we have sufficient data used for training, we decided to drop those rows that has missing values in either "BodyType", "DriveType", "zipcode" and "model" categorical feature.

We need to check for anomaly in data. We find that feature "Year" has some invalid data such as 2014000 and 1008, it is most likely because of data entry errors. So we need to filter out "Year" data with value 1900-2024. We find some extremely high "Mileage" values (e.g., 1e9), which are also likely anomalies since typically a reasonable range for most cars should be 0-300000. We only maintain "Mileage" data within range 0-600000. Some rows have "pricesold" value 0, which may indicate erroneous or missing values, we also need to eliminate them. Some "zipcode" value in the dataset are non-numeric or incomplete (e.g., 947** or VW), we need to apply regex expressions to only retained numeric ones.

Columns like *"Make, BodyType, DriveType"* may have inconsistent capitalization or spelling variations (e.g., "CHROVELET" vs "chrovelet"), but they stand for the same meaning, so we need to convert these category features to lowercase and standardize categories in order for latter one-hot encodings.

We need to explore the correlations between each pair of numerical features to check whether they have strong correlations. If 2 features are found to be have strong correlations, we either combine these 2 features or delete one of them, to avoid redundancy. Correlation Coefficient is derived by

$$\rho_{X,Y} = \frac{Cov(x,y)}{\sigma_x \sigma_y}$$

We derive the correlation matrix of the categorical features and plot out a heatmap visualization as in **Figure 2**. It can be observed that the features are not-related as their Correlation Coefficient are all close to 0.

We need to standardized the numerical data. Features like *"Mileage,Year,NumCylinders"* have different scales. Without scaling, features with larger values may dominate the model. The StandardScaler transforms each feature x using:

$$\frac{x - \mu}{\sigma}$$

Standardizing data helps models converge faster and make better predictions by ensuring all features contribute equally.

For categorical data, we need to do one-hot encoding before feeding them into machine learning model training process. One-hot encoding converts categorical variables into a numerical format that machine learning algorithms can process. It's particularly useful for models that require numerical input only.

2.3 Data Visualization

We first visualize and do exploration within a single feature. In **Figure 3**, We plot a pie chart that explore the used car sales distribution of different brand ("Make") and sort out the top 10 brands. Among all available sales data, 18% of sales are from Ford, 16.2% are from Chrovelet, Toyota account for 5.8%. American and European brand cars have good sales record. **Figure 4** and **Figure 5** are boxplot and histogram that displays the distribution of price that the used cars are sold. From the boxplot, we can observe the data for "pricesold" has a large number of "outliers". Most values are concentrated on the lower end, between 0 and 25000, but a significant number of extreme values (outliers) extend far to the right till 400000. The histogram confirms the skewness seen in the boxplot. The distribution is highly right-skewed with very few occurrences of higher prices (e.g., above 100,000). We further investigated those data with extremely high sold price, but we are unable to conclude whether they are anomalies or they are indeed actual prices, since data with high "pricesold" indeed comes from luxury car brands such as Rolls-Royce, Porsche.

The relationship between features and the target variable "pricesold" are then further investigated. The violin plot in **Figure 6** provides insight into the distribution of "pricesold" across the top 10 body types (and "Other"), in which wider areas of the violin plot represent where the majority of prices are concentrated. Narrower areas show fewer data points. Most "BodyType" have price ranges concentrated at the lower end, which are consistent with our daily experience. SUVs, Coupes and Convertibles tend to have wider price ranges, including a higher proportions of more expensive vehicles. Relationship between "pricesold" and "Mileage" are displayed in scatter plot in **Figure 7** and **Figure 8**. There is a clear negative relationship between "Mileage" and price. As "Mileage" increases, the price sold tends to decrease. At very high

mileage (above 200,000), prices cluster near the lower end (close to 0–$10,000), suggesting cars with very high mileage are generally sold for much lower prices. **Figure 8** adds a categorical dimension (”DriveType”) as a hue to the relationship between mileage and price, in which we observe that ”rwd”, ”awd”, and ”4wd” dominate the lower mileage range. High mileage vehicles appear to be more common in front-wheel drive and rear-wheel drive categories.

We explore the car sales density across geological locations in **Figure 9**, by incorporating the service library ”pgeocode” to map ”zipcode” values into geographical coordinates. We then plot sales density using a Folium map with markers or heatmaps. Green and Yellow Areas represent regions with a higher density of car sales. The brighter the color, the more concentrated the sales, while blue or purple Areas indicate regions with a lower density of car sales. We can find that Areas around major cities like Los Angeles, New York, and parts of Texas show higher car sales activity. These are the areas with larger population density and might have higher transportation demands. In contrast, large parts of the Mountain West such as Wyoming, Montana, and Idaho and certain rural regions show very low car sales activity. These regions are less populated, which likely contributes to lower sales densities. In general, Car sales activity is strongest along coasts like California and the Eastern Seaboard. There is a noticeable decline in density as we move inland to less urbanized areas. The source code (see **Section 4 Note**) has better visualization of the density map than the figures, in which we are able to zoom in the geographical areas for better visualization on the density heatmap (in colors). **Figure 10** and **Figure 11** offers an alternative geographical density visualization implemented from Plotly, it provides better interactions, in which we are able to click on a map using latitude and longitude and see the sales density at a particular zipcode in the map.

3 Model Prediction

3.1 Models

After the data are preprocessed, we begin to construct the model that make predictions on the ”pricesold” of the used cars based on the features involved. We will split the dataset into training dataset and evaluation dataset, and evaluate the model trained on the training dataset using the evaluation dataset. The evaluation metric we used here is Root Mean Square Error (RMSE), which is a prevalent evaluation metrics for regression problems. RMSE is estimated as

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\epsilon_i^2}$$

where $\epsilon_i = y_{pred} - y_{true}$.

The first algorithm we used in fitting the model is **Linear Regression**, which is a traditional algorithm for regression problems. Linear Regression is a supervised learning algorithm used for predicting a continuous target y, based on multiple features $x_1, x_2, x_3, ...x_n$. It models the relationship between the features and the target as a linear equation:

$$y = w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n + b$$

Numerical features are standardized using StandardScaler to ensure all features have a mean of 0 and unit variance. Categorical features (e.g., *”Make, Model, DriveType”*) are converted into numerical format using OneHotEncoder, as mentioned in **Section 2.2**, before fitting in the model. The best parameters of the Linear Regression model in this problem is $\{$*”regressor__copy_X”: True, ”regressor__fit_intercept”: True, ”regressor__n_jobs”: None*$\}$

The second model is the **XGBoost** algorithm, which is a powerful, scalable implementation of gradient boosting that builds an ensemble of decision trees to solve regression and classification tasks. Boosting combines weak learners (decision trees) iteratively, where each tree corrects the errors of the previous ones. The final prediction is the weighted sum of all tree outputs. Each tree is trained to minimize a loss function (e.g., Mean Squared Error for regression) by using the gradient of the loss with respect to the predictions.

At each iteration:

$$NewPrediction = PreviousPrediction - \eta\nabla L$$

where η is the learning rate and ∇L is the gradient of the loss function. XGBoost starts with a simple model, then in each iterations, it fits a tree to the residual errors, update predictions using the new tree’s output scaled by the learning rate. It stops after a fixed number of trees (n_estimators) or when the performance improvement becomes negligible. XGBoost is powerful in this problem because it can

handle large datasets and features with high cardinality, such as the categorical features "zipcode", "DriveType" with many unique values. The best parameters of the XGBoost model in this problem is {*"regressor__learning_rate": 0.1, "regressor__max_depth": 7, "regressor__n_estimators": 300*}

The third model is a **Neural Network** model. It is a fully connected feedforward neural network. Preprocessed data after onehot encoding and standardscaling are fed to the neural network. In the Input Layer: the input size corresponds to the number of preprocessed features after one-hot encoding and scaling. Each input feature is fed into the first Hidden Layer. Layer 1 of the Hidden Layer has 128 neurons, with ReLU (Rectified Linear Unit) as its activation function, which introduces non-linearity by outputting $max(0, x)$. It was then followed by a Dropout layer with a rate of 0.2 to prevent overfitting. Layer 2 has 64 neurons, with ReLU as activation, and followed by another Dropout layer with a rate of 0.2. Layer 3 has 32 neurons, ReLU activation and then connected to the Output Layer. The Output Layer is a 1 neuron to output the final predicted value. Unlike linear regression or simpler models, the neural network can capture non-linear patterns in the data. It can efficiently process large numbers of features, such as those created after one-hot encoding categorical variables like *"zipcode, Make, Model"*.

3.2 Prediction

We measure the RMSE values on the training dataset and evaluation dataset for the above 3 models and present those values in **Table 1**.

Model	LR	XGBoost	NN
$RMSE_{train}$	4105.06	5488.24	2279.96
$RMSE_{eval}$	10348.00	7137.04	8010.08

Table 1: RMSE values on training dataset and evaluation dataset

The table shows that XGBoost model performs best in predicting used car price under the current parameter setting. Even though Neural Network has the lowest RMSE in training set, it has larger RMSE at the evaluation dataset, indicating that overfitting may occur in the Neural Network. This might also because XGBoost performs exceptionally well on small-to-medium-sized datasets as decision trees are robust and require less data to learn patterns effectively, while Neural Network typically require large amounts of data to generalize well. Since the current dataset is not at large size, Neural Network may overfit to the training data, which can be observed from the lower $RMSE_{train}$ but higher evaluation $RMSE_{eval}$ in the model output.

4 Note

Please note that:

- The source code and output files are in github repository: https://github.com/yangbright-2001/ece225A-project

References

[1] Data Source: US used car sales data: https://www.kaggle.com/datasets/tsaustin/us-used-car-sales-data/data

[2] eBay website - nonbinding policy: https://www.ebay.com/help/policies/rules-policies-buyers/nonbinding-bid-policy?id=4228

5 Appendix

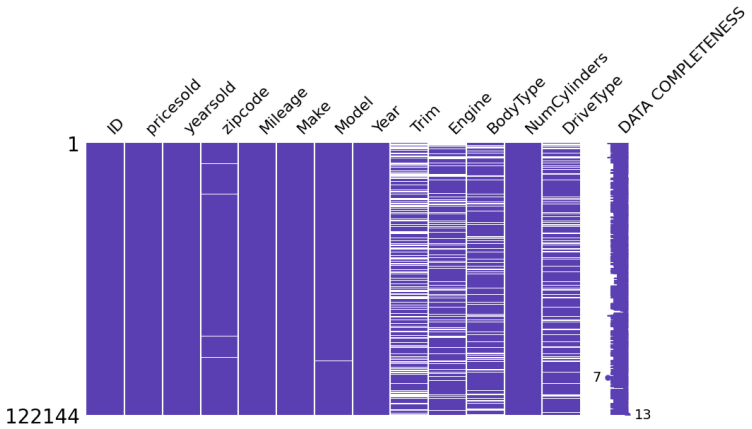

Figure 1: Data missing visualization of different features

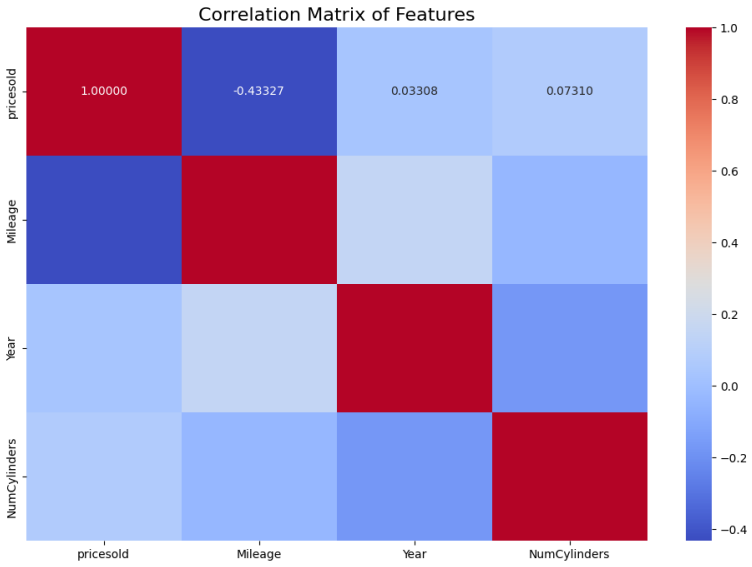

Figure 2: Heatmap of numerical feature correlations

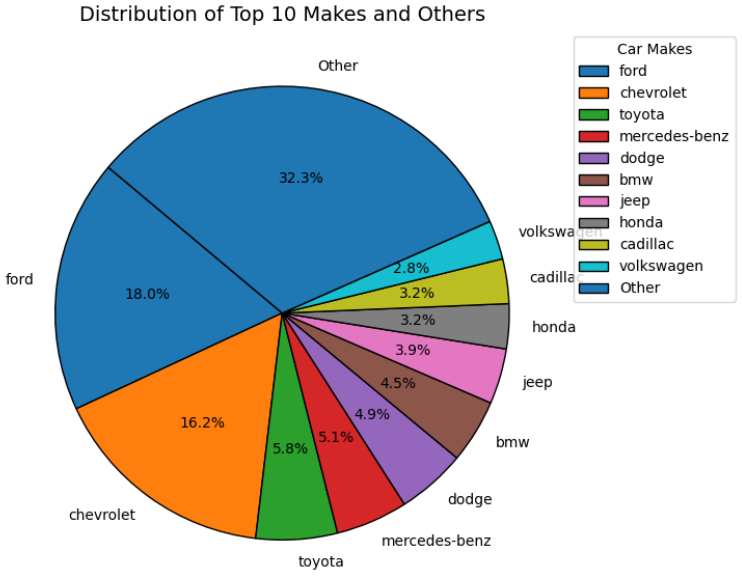

Figure 3: Pie chart of Top10 Makes

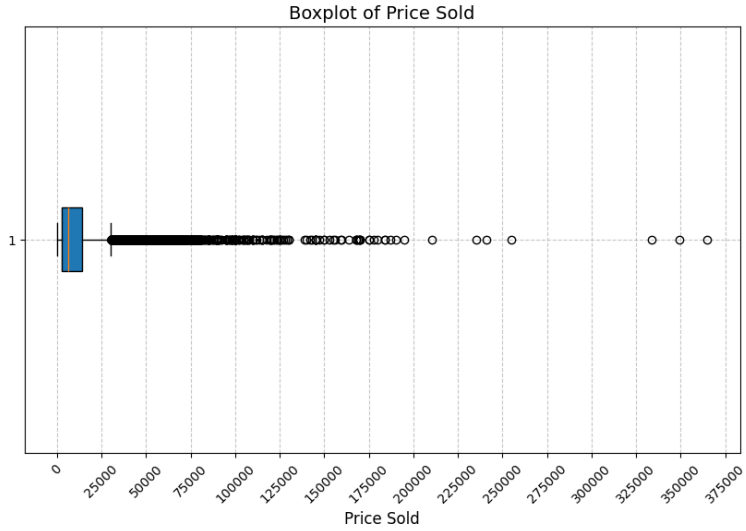

Figure 4: Box plot of pricesold distribution

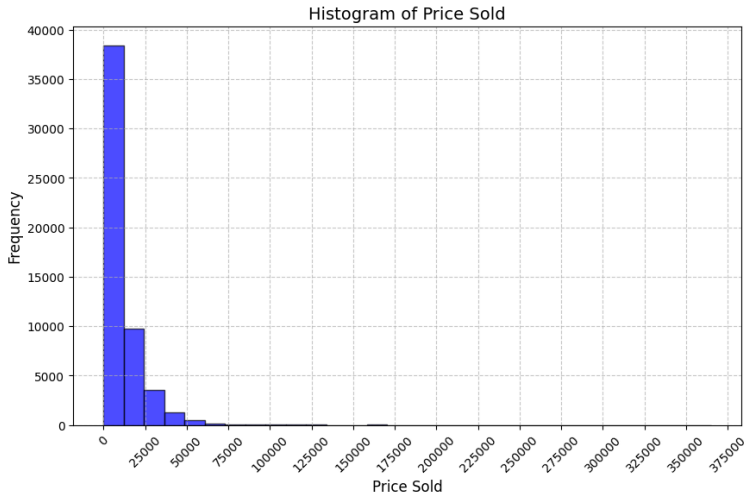

Figure 5: Histogram of pricesold distribution

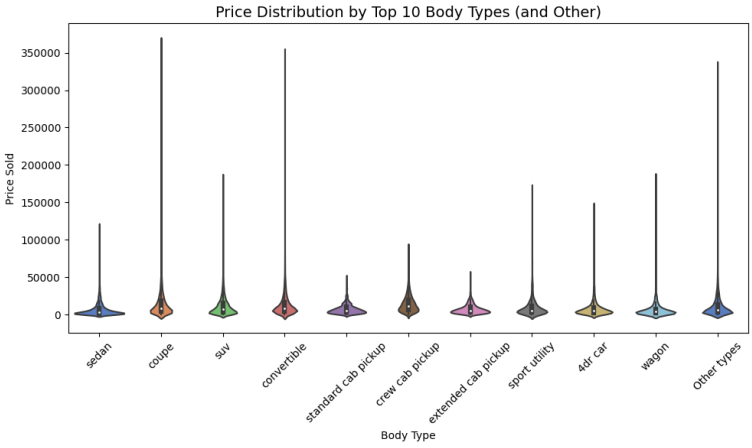

Figure 6: Violin plot of BodyType and pricesold

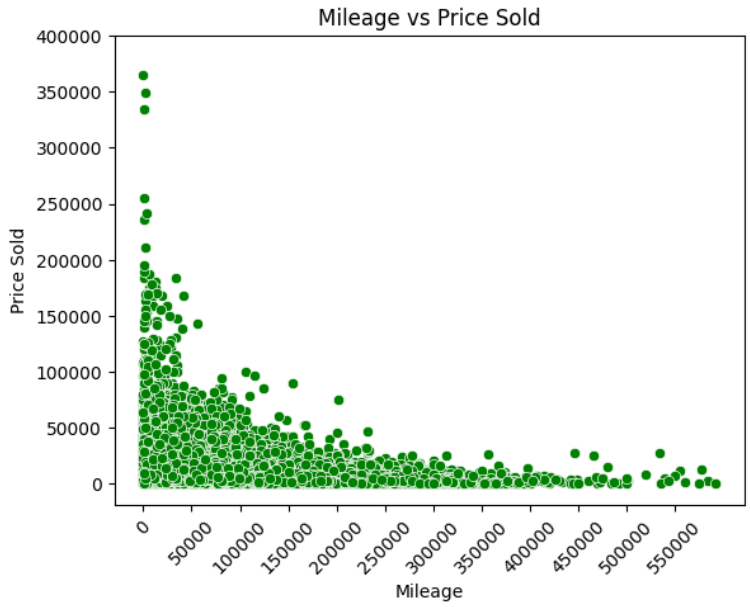

Figure 7: Scatter plot of mileage and pricesold

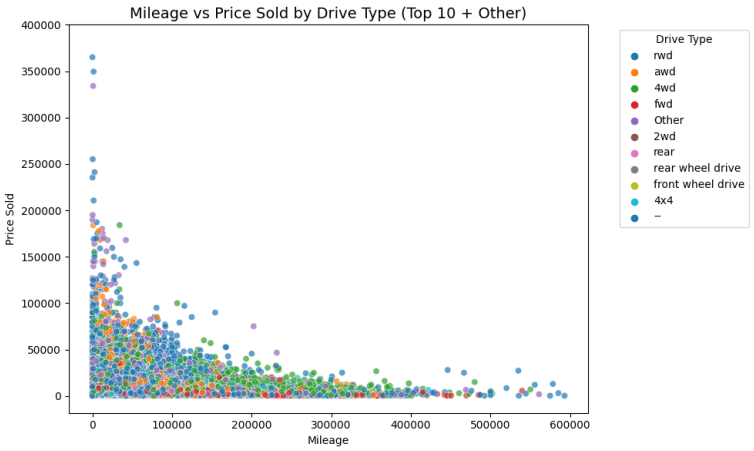

Figure 8: Scatter plot of mileage and pricesold with DriveType as hue

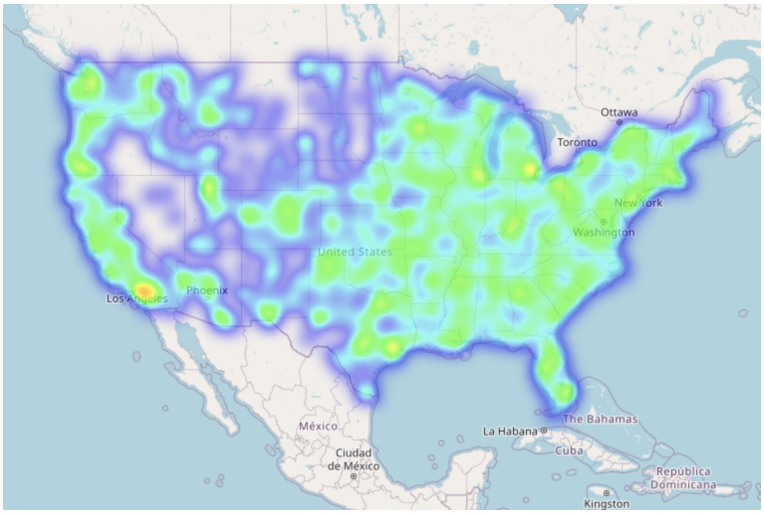

Figure 9: Car sales by location - Folium heatmap

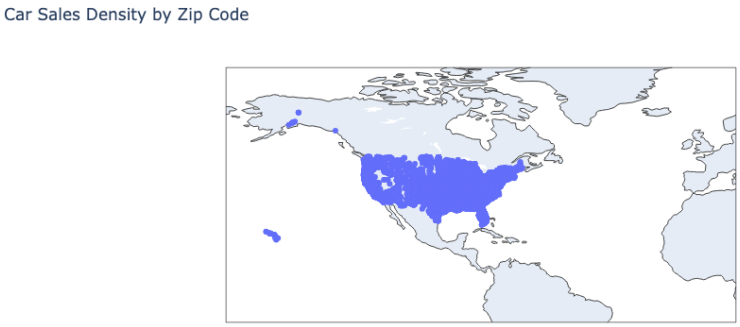

Figure 10: Car sales by location - Plotly interactive map

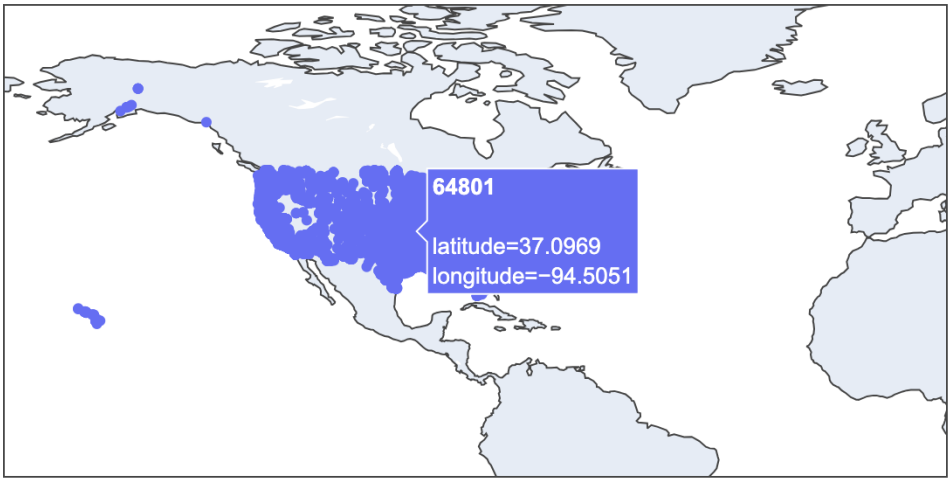

Figure 11: Car sales by location - Plotly interactive map with geological