CASE STUDY

Car Sales Prediction

Artificial Neural Network

Author: Sameer Hate

Date: 11-05-2025

Table of Contents

Introduction

General

In the modern automotive industry, leveraging data-driven approaches to understand and anticipate customer behavior has become increasingly important. The ability to accurately estimate a customer's potential car purchase amount can offer significant advantages in optimizing sales strategies, pricing models, and customer engagement efforts.

This study focuses on building a predictive model using machine learning techniques to forecast the car purchase amount based on a range of customer attributes. The dataset utilized includes demographic and financial information of many customers. These features serve as the independent variables, while the target variable is the total dollar amount a customer is likely to spend on a vehicle.

In this study, an Artificial Neural Network (ANN) is developed and trained to predict the car purchase amount based on various customer attributes. The objective is to leverage the ANN's ability to model complex, non-linear relationships between inputs and outputs to achieve high predictive accuracy. This approach not only demonstrates the practical application of deep learning in sales forecasting but also provides valuable insights into how demographic and financial characteristics influence purchasing behavior. The results of this project hold potential for enhancing customer segmentation, enabling targeted marketing, and supporting personalized pricing strategies within the automotive sales sector.

Author's Note

This case study presented in this work has been independently completed by me. This project serves as a medium to deepen my understanding of Data Science concepts and enhance my practical skills in applying machine learning and data analysis techniques to real-world problems.It may be prone to errors but I try my best to eliminate them as much as I can. In case you identify any such error or have a more optimal technique to solve this problem, your feedback and suggestions are always welcome as this will eventually help me strengthen my concepts.

Problem Statement & Expected Results

Objective

To develop a predictive model that accurately estimates the Car purchase amount a customer is likely to pay, using their demographic and financial attributes.

Goals

- To collect and analyze customer demographic and financial data relevant to car purchase behavior.
- To design and implement an Artificial Neural Network (ANN) capable of modeling non-linear relationships between features and purchase amount.
- To evaluate the model's performance using metrics such as Mean Absolute Error (MAE) and R^2 Score.
- To gain insights into how factors like income, net worth, and credit card debt influence car purchase decisions.
- To improve prediction accuracy through feature scaling and tuning of ANN architecture.

Expected Outcomes

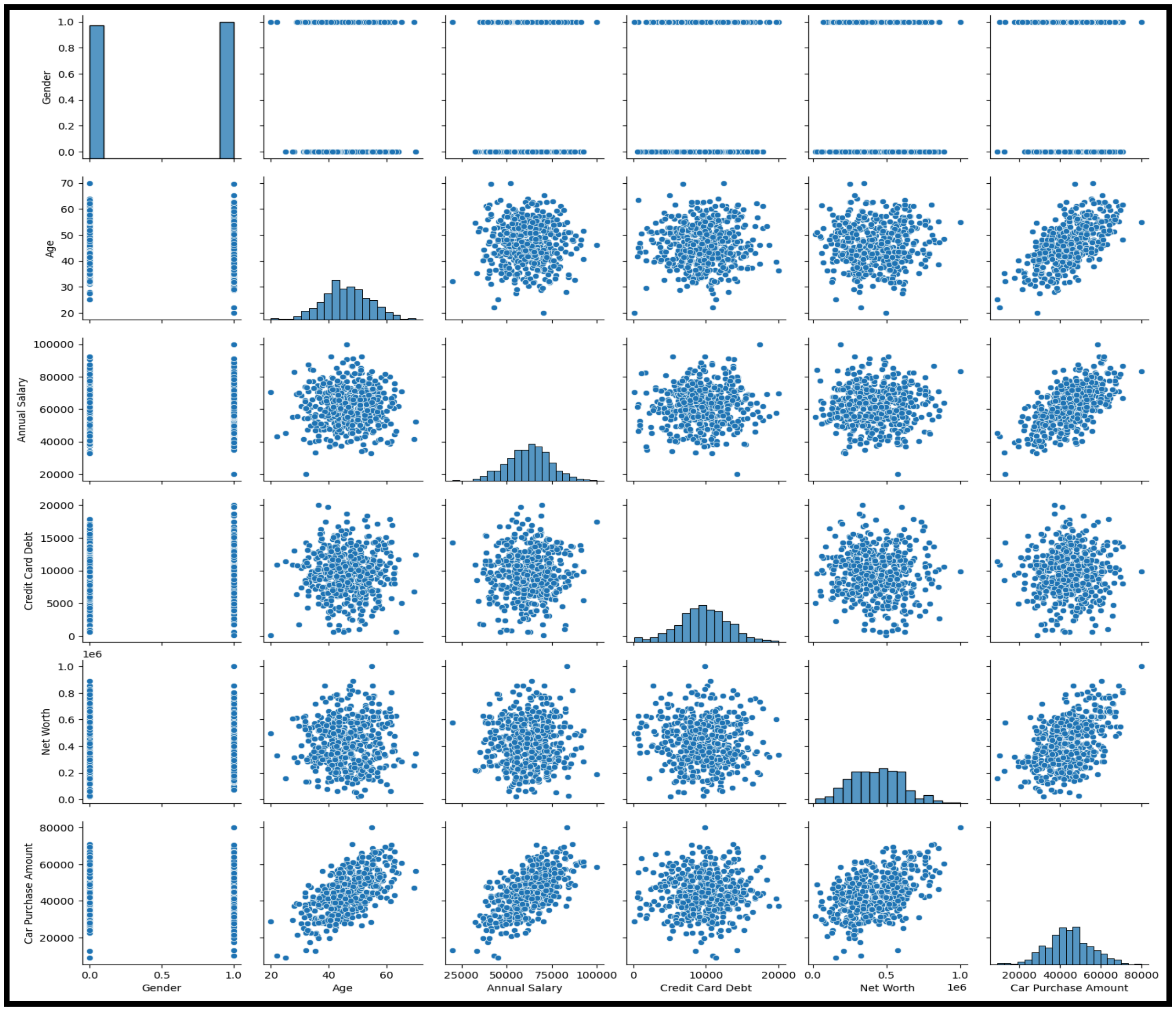
The expected outcome of this project is a trained Artificial Neural Network (ANN) model capable of accurately predicting the car purchase amount a customer is likely to spend, based on their demographic and financial data. The model should generalize well to new, unseen data and demonstrate strong performance metrics such as low Mean Absolute Error (MAE) and high R^2 Score.

Dataset

Description

The dataset used in this study consists of 500 customer records, each containing demographic and financial information relevant to car purchasing behavior. It includes attributes such as customer name, email, country of residence, gender, age, annual salary, credit card debt, and net worth. The target variable is the car purchase amount, representing the total dollar value a customer is likely to spend on purchasing a car. The dataset reflects a diverse range of financial profiles, with ages ranging from 20 to 70 years, annual salaries between $20,000 and $100,000, and net worth values up to $1 million. This rich variety of inputs enables the model to learn complex patterns and relationships that influence a customer's purchasing capacity in the automotive sector.

Link to the Dataset: Click Here



Approach

- #IMPORTING LIBRARIES : This cell imports all the necessary libraries required for building, training, and evaluating an Artificial Neural Network (ANN) for regression. These imports collectively support the end-to-end machine learning pipeline used in this project.Specifically:
    - pandas and numpy are used for data loading, manipulation, and numerical operations.
    - tensorflow provides the deep learning framework for constructing and training the ANN.
    - matplotlib.pyplot and seaborn are used for data visualization and analysis.
    - ColumnTransformer and OneHotEncoder from sklearn.compose and sklearn.preprocessing handle categorical data encoding.
    - MinMaxScaler is applied to normalize feature values for better ANN performance.
    - train_test_split is used to divide the dataset into training and testing sets.
    - mean_absolute_error and r2_score are evaluation metrics to measure model accuracy and goodness of fit.

- #PREPARING THE DATASET & PRINTING : This cell loads the dataset into the environment using the pandas library. An Encoding of 'ISO-8859-1' is mentioned to ensure compatibility with special character encodings in the file.

- #VISUALIZING THE DATASET : This step involves visualizing the relationships between variables in the dataset using a pair plot generated with the seaborn library. The sns.pairplot() function creates a grid of scatter plots for all numerical feature combinations, along with histograms on the diagonals. This visualization helps in identifying correlations, trends, clusters, and potential outliers among the features. It serves as an effective exploratory data analysis (EDA) tool to understand the structure and distribution of the data before proceeding with preprocessing and model training.

- #CLEANING THE DATASET BY DROPPING UNNECESSARY COLUMNS: In this step, the dataset is cleaned by removing non-informative and target-related columns. Specifically, the columns 'Customer Name' and 'Customer e-mail' are excluded from the feature set (X) as they are identifiers and do not contribute to the predictive modeling process. The target variable, 'Car Purchase Amount', is also separated from the dataset and assigned to y, which will serve as the dependent variable during model training. Additionally, the shape of X and y is checked to ensure the data has been correctly partitioned into features and target variables.

- #ONE HOT ENCODING CATEGORICAL DATA : Here we are handling the categorical variable ‘Country’ using One-Hot Encoding, a common preprocessing technique to convert categorical data into a numerical format suitable for machine learning models.he ColumnTransformer from scikit-learn is used to apply OneHotEncoder specifically to the 'Country' column, while the remainder='passthrough' argument ensures that all other columns remain unchanged. The sparse_output=False parameter instructs the encoder to return a dense array instead of a sparse matrix. As a result of encoding a large number of unique country values, the number of input features increases significantly—from 6 to 216 columns—reflecting the expanded dimensionality. The transformed feature set is reassigned to X, now consisting solely of numerical values, making it ready for subsequent scaling and model training.

- #APPLYING FEATURE SCALING TO NORMALIZE THE VALUES: In this step, feature scaling is applied to normalize both the input features (X) and the target variable (y). The MinMaxScaler from scikit-learn is used to scale the values to a standard range between 0 and 1. This transformation is particularly important when training an Artificial Neural Network (ANN), as it ensures that all input features contribute proportionally to the model’s learning process and prevents features with larger magnitudes from dominating the optimization. Two separate scaler instances are used—sc_x for the independent variables and sc_y for the dependent variable—to preserve the integrity of the scaling operation. The target vector y is reshaped into a 2D array before scaling, in accordance with the scaler’s input requirements.

- #VISUALIZING THE DATASET AFTER FEATURE SCALING : Printing the scaled columns of the dataset just for verification purposes.

- #SPLITTING THE DATASET INTO TRAINING AND TESTING SETS : In this step, the dataset is divided into training and testing subsets using the train_test_split function from scikit-learn. Specifically, 80% of the data is allocated for training the Artificial Neural Network (ANN), while the remaining 20% is reserved for evaluating its performance on unseen data. This split ensures that the model can learn patterns from the majority of the data while being validated against a separate subset to assess its generalization capability. The random_state parameter is set to 0 to ensure the split is reproducible across multiple runs, allowing for consistent evaluation and debugging.

- #BUILDING THE ARTIFICIAL NEURAL NETWORK(ANN) MODEL : This step marks the beginning of the Artificial Neural Network (ANN) construction. Using TensorFlow’s keras API, a sequential model is initialized through tf.keras.models.Sequential(). This model serves as a linear stack of layers, where each layer has exactly one input tensor and one output tensor. At this stage, no layers have been added yet — this simply creates an empty ANN architecture to which layers will be added sequentially in the subsequent steps.

- #ADDING THE INPUT LAYER AND FIRST HIDDEN LAYER : In this step, the input layer and the first hidden layer of the Artificial Neural Network (ANN) are added using the Dense layer from TensorFlow’s Keras API. The layer consists of 75 neurons (units) and uses the ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to help the network learn complex patterns. The input_dim=216 argument specifies that the input layer expects 216 features, which corresponds to the number of columns in the preprocessed dataset after one-hot encoding and scaling. This forms the first trainable layer of the model.

- #ADDING THE SECOND & THIRD HIDDEN LAYER : These code cells add the second and third hidden layer to the Artificial Neural Network (ANN) using Keras’ Dense layer. Like the first hidden layer, they contain 75 neurons and use the ReLU (Rectified Linear Unit) activation function to introduce non-linearity. Since this layer follows the first, the input dimension does not need to be explicitly specified—Keras automatically infers it based on the output of the preceding layer. The second and third hidden layer helps the network learn more abstract and higher-order features from the data.

- #ADDING THE OUTPUT LAYER : In this step, the output layer is added to the Artificial Neural Network (ANN) using a Dense layer with a single neuron (units=1). Since the task is a regression problem, the activation function is set to 'linear', allowing the model to predict continuous numerical values without applying any transformation. This layer outputs the predicted car purchase amount based on the learned patterns from the input features through the hidden layers.

- #CHECKING THE MODEL SUMMARY : This line calls the summary() method on the constructed ANN model to display a detailed summary of its architecture. The output includes information about each layer, the number of neurons, the activation functions used, the number of trainable parameters, and the total number of parameters in the model. Reviewing the model summary helps verify the structure of the network and ensures that all layers are configured as intended before proceeding to model compilation and training.

- #COMPILING THE ANN : In this step, the Artificial Neural Network (ANN) is compiled using the compile() method provided by Keras. The optimizer is set to 'adam', an adaptive learning rate optimization algorithm known for its efficiency and suitability for deep learning tasks. The loss function used is 'mean_squared_error', which is appropriate for regression problems where the goal is to minimize the average squared difference between predicted and actual values. Additionally, Mean Absolute Error (MAE) is specified as a performance metric to monitor during training, offering an interpretable measure of the model's average prediction error in absolute terms.

- #TRAINING THE ANN ON THE TRAINING SET : Here the Artificial Neural Network (ANN) is trained using the fit() method. The model is trained on the preprocessed training data (X_train, y_train) with a batch size of 10, meaning the weights are updated after every 10 samples. The training is conducted over 300 epochs, allowing the model to iteratively learn and refine its weights. The verbose=1 parameter ensures that the training progress, including loss and metric values for each epoch, is displayed in the console. A validation split of 0.2 is specified, meaning 20% of the training data is set aside for validation during training. This helps monitor the model's performance on unseen data and detect potential overfitting.

- #EVALUATING THE MODEL: This step accesses the training history of the model by retrieving the keys from the history object returned by the fit() method. Specifically, output.history.keys() displays the names of all metrics and loss values recorded during each epoch of training. These typically include training loss, validation loss, and any additional metrics such as Mean Absolute Error (MAE). This information is essential for analyzing the learning behavior of the model over time and can be used to generate plots that visualize training and validation performance across epochs.

- #PLOTTING ALL THE VALUES RETURNED BY THE FIT METHOD : This code cell visualizes the model's performance during training by plotting the training loss and validation loss recorded over each epoch. Using the history object returned by the fit() method, output.history['loss'] represents the model's loss on the training data, while output.history['val_loss'] represents the loss on the validation set. These losses are plotted on the same graph to facilitate comparison. The plot includes labeled axes and a legend, providing a clear view of how the model's learning evolved over time. This visualization is useful for diagnosing issues such as overfitting (if validation loss increases while training loss decreases) or underfitting (if both losses remain high).

- #CALCULATING THE MEAN ABSOLUTE ERROR AND R2 VALUE : In this step, the trained Artificial Neural Network (ANN) model is evaluated on the test dataset using two key regression metrics: Mean Absolute Error (MAE) and R^2 Score. First, predictions are generated using the predict() method on X_test. Since the target variable was previously scaled using MinMaxScaler, both the predicted and actual values (y_pred and y_test) are inverse-transformed using sc_y.inverse_transform() to restore their original scale. The Mean Absolute Error measures the average absolute difference between the predicted and actual values, offering a direct interpretation in the original units (e.g., dollars). The R^2 Score (coefficient of determination) quantifies how well the model explains the variance in the data, with a value closer to 1 indicating a better fit. The computed values are printed to assess the model's performance.

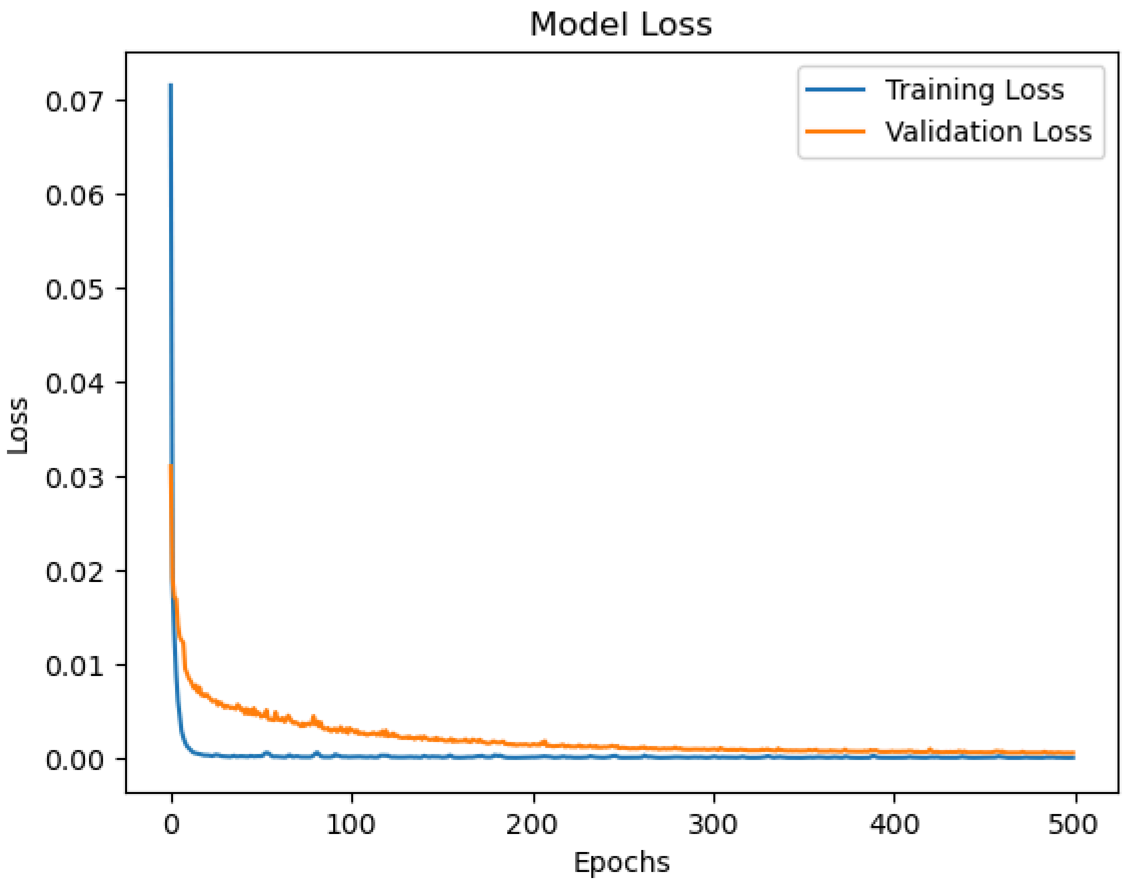
- #PREDICTING NEW VALUE : This step demonstrates the use of the trained ANN model to predict the car purchase amount for a new, previously unseen customer profile. A single-row DataFrame is constructed with values representing a hypothetical customer from Brazil, including demographic and financial information such as gender, age, annual salary, credit card debt, and net worth. The 'Country' column is encoded using the same ColumnTransformer (ct) that was applied during training to ensure consistency in feature representation. The resulting encoded input is then scaled using the previously fitted MinMaxScaler (sc_x) to match the training data scale. The ANN model then predicts the scaled output value using predict(). Since the model output is also scaled, the prediction is inverse-transformed using sc_y to return the predicted car purchase amount in its original dollar scale. Finally, the expected purchase amount is printed, providing a practical demonstration of the model's application in real-world scenarios.

<u>Results</u>

<u>Overall</u>

After experimenting with various neural network configurations as seen below, the best-performing model consisted of four hidden layers with 32, 16, 32, and 32 units, respectively. This architecture achieved a Mean Absolute Error (MAE) of **1005.44** and an R^2 score of **0.9763**, indicating that the model explains approximately **97.63%** of the variance in the target variable. These results reflect a highly accurate model for predicting car purchase amounts based on customer demographic and financial attributes.

**It is important to note that due to the inherent randomness in model initialization and training (e.g., weight initialization and data shuffling), and because random_state=0 was used without fully fixing all random seeds, exact replication of these results may vary across runs unless strict reproducibility measures are applied.*



Training and Validation Loss Curves

The loss curve illustrates the training and validation loss trends of the Artificial Neural Network (ANN) model over 500 epochs. Initially, both training and validation loss decrease sharply, indicating that the model is effectively learning the underlying patterns in the data. As training progresses, the losses continue to decline and gradually stabilize, demonstrating smooth and consistent convergence. Importantly, the validation loss closely follows the training loss throughout the training process, with no significant divergence. This suggests that the model generalizes well to unseen data and is not overfitting. Overall, the plot reflects a well-behaved learning process, with the model successfully minimizing error on both the training and validation sets over time.

Specific Prediction

The following values were submitted to the model in order to make a prediction,

- Country - Brazil
- Gender - 1(Male)
- Age - 57
- Annual Salary - $59,000
- Credit Card Debt - $5,400
- Net Worth - $5,60,000

Model Prediction : Expected Purchase Amount: **$55,086.55**

A customer from Brazil, 57 years old, with an annual salary of $59,000, credit card debt of $5,400, and a net worth of $560,000. The trained ANN model predicted the expected car purchase amount to be approximately $55,086.55.

Given the model's performance metrics—R^2 score of 0.9763 and Mean Absolute Error (MAE) of $1,005.44—this prediction can be considered highly reliable. The R^2 value indicates that the model explains 97.63% of the variance in the car purchase amounts, suggesting that the prediction falls well within the range of expected accuracy. Additionally, the MAE implies that on average, the model's predictions deviate from the actual values by approximately $1,005, making a prediction of $55,086.55 reasonably accurate within that expected error margin.

References

Units in 1st Layer	Units in 2nd Layer	Units in 3rd Layer	Batch Size	Epochs	MAE	R^2 Value
64	128	64	5	500	2240.73	0.93
128	128	64	5	500	2914.1	0.89
64	128	128	5	500	2028.41	0.94
64	128	256	5	500	2779.59	0.89
64	128	128	5	700	2998.06	0.89
128	128	128	5	500	2763.25	0.89
128	256	128	5	500	2854.47	0.89
128	256	256	5	500	3067.79	0.87

3 Layers - Trial Run

Units in 1st Layer	Units in 2nd Layer	Units in 3rd Layer	Units in 4th Layer	Batch Size	Epochs	MAE	R^2 Value
32	32	32	32	5	500	1739.57	0.95
16	32	32	32	5	500	1867.27	0.93
32	16	32	32	5	500	1005.44	0.97
32	32	16	32	5	500	2056.38	0.93
32	32	32	16	5	500	1518.39	0.95
8	8	8	8	5	500	1597.91	0.93
8	16	8	8	500	500	1448.14	0.94

4 Layers - Trial Run

Units in 1st Layer	Units in 2nd Layer	Units in 3rd Layer	Units in 4th Layer	Units in 5th Layer	Batch Size	Epochs	MAE	R^2 Value
32	32	32	32	32	5	500	2405.02	0.92
16	32	32	32	32	5	500	1554.75	0.94
32	16	32	32	32	5	500	1551.59	0.96
32	32	16	32	32	5	500	1863.2	0.94
32	32	32	16	32	5	500	2523.89	0.91
32	32	32	32	16	5	500	1548.34	0.95

5 Layers - Trial Run

GitHub Repository - Click Here

****