Project 2: Insurance Claim Prediction

(First discussion: Apr 12; Deadline: May 17)

The goal of this project is to implement and compare different models for insurance frequency claim prediction on real-life data from the French motor third party liability dataset (see file `freMTPL2freq.csv`), which comprises $m = 678{,}007$ car insurance policies.
Below is an overview of the data:

Feature name	**Feature description**	**Feature type**[1] **and range**
`VehPower`	Car power	Discrete, values in $\{4, 5, \ldots, 15\}$
`VehAge`	Car age in years	Discrete, values in $\{0, 1, \ldots, 100\}$
`DrivAge`	Driver's age in years	Discrete, values in $\{18, 19, \ldots, 100\}$
`BonusMalus`	Driver's bonus-malus level	Discrete, starts at 100, decreases if no accidents, increases otherwise
`VehBrand`	Car brand	Categorical, values in {'B1', 'B2', ..., 'B14'}
`VehGas`	Car fuel type	Categorical, values in $\{\text{diesel}, \text{regular}\}$
`Density`	Population density (inhabitants per km^2) at driver's place of residence	Discrete, from 1 to 27,000
`Region`	Driver's region of residence	Categorical, values in $\{\text{R11}, \ldots, \text{R94}\}$

Label name	**Label description**	**Label type**
`Exposure`	Duration of insurance policy in years	Continuous, values in $[0, 1]$
`ClaimNb`	Number of insurance claims filed during policy's lifetime	Discrete, values in $\{0, 1, \ldots, 5\}$

1. **Data Exploration.** Get a first impression of the datasets by plotting the features and the claim frequency (defined as $y^i = \texttt{ClaimNb}_i/\texttt{Exposure}_i$) label (e.g., using pairplots). Include additional data analysis that you find useful. Describe what you see.

2. **Poisson GLM.**

   We first fit a Poisson Generalized Linear Model (GLM), which is commonly used for insurance frequency claim prediction.

   Let $\texttt{ClaimNb}_i$ be the number of claims and $\texttt{Exposure}_i$ be the duration in years of the i-th policy. We want to predict the claim frequency $y^i = \texttt{ClaimNb}_i/\texttt{Exposure}_i$ given a training set of insurance policy features $D = \{(x^i, y^i) \in \mathbb{R}^d \times \mathbb{R}_+,\ 1 \le i \le m\}$.

   Under the Poisson GLM we assume that $y^i \cdot \texttt{Exposure}_i \sim \text{Poisson}(\lambda^i \cdot \texttt{Exposure}_i)$ with mean parameter λ^i of the form

$$\lambda^i = \exp\left(\sum_{j=1}^{d} \theta_j x_j^i + \theta_0\right), \tag{1}$$

[1] A *discrete* feature is a feature that takes finitely many values in an ordered set, such as $\{1, 2, \ldots, 5\}$. A *continuous* feature takes values in a dense interval, such as $[0, 1]$. A *categorical feature* is a feature that takes finitely many values $\{v_1, \ldots, v_k\}$, called *levels*, which are not assumed to be in an ordered relationship.

for some regression coefficient $\theta = (\theta_0, \theta_1, \ldots, \theta_d) \in \mathbb{R}^{d+1}$ to be estimated.

The model is fitted by minimizing the following loss (more formally known as the *exposure-weighted Poisson deviance*):

$$\mathcal{L}(D, \hat{\theta}) = \frac{1}{\sum_{i=1}^{m} \texttt{Exposure}_i} \sum_{i=1}^{m} \texttt{Exposure}_i \cdot \ell(\hat{\lambda}^i, y^i), \tag{2}$$

where $\hat{\lambda}^i = \exp\left(\sum_{j=1}^{d} \hat{\theta}_j x_j^i + \hat{\theta}_0\right)$ is the estimate of each policy's mean frequency and

$$\ell(\hat{\lambda}, y) = 2\left(\hat{\lambda} - y - y \log \hat{\lambda} + y \log y\right),$$

with the convention that $x \log(x) = 0$ if $x = 0$.

(a) Given the functional dependence in (1), when training a Poisson GLM model it is often necessary to transform the dataset features to guarantee a good fit. This procedure is known as *feature engineering* or *feature pre-processing* and typically requires expert knowledge and extensive data analysis.

Pre-process the following dataset features as indicated:

- `VehPower`: transform to log(`VehPower`),
- `VehAge`: convert to categorical feature with levels $[0, 6), [6, 13), [13, \infty)$.
- `DrivAge`: transform to log(`DrivAge`).
- `BonusMalus`: transform to log(`BonusMalus`).
- `Density`: transform to log(`Density`).

(b) Perform a 90%-10% train-test split and fit a Poisson GLM. You may use the implementation available at `sklearn.linear_model.PoissonRegressor`. Remember to instantiate the model with `alpha` equal to zero (i.e. without regularization) and to minimize the *weighted* Poisson deviance by fitting the model with the `Exposure` as value for the argument `sample_weight`. Report the weighted Mean Absolute Error (MAE), weighted Mean Squared Error (MSE) and the loss $\mathcal{L}$ on train and test sets.

Remark: remember to standardize all continuous and discrete features and to transform all categorical features using one-hot encoding. Beware of data normalization issues!

(c) Add the following engineered features[2] to the training set:

$$(\texttt{DrivAge},\ \texttt{DrivAge}^2,\ \texttt{BonusMalus} \cdot \texttt{DrivAge},\ \texttt{BonusMalus} \cdot \texttt{DrivAge}^2),$$

and verify that the Poisson GLM achieves a slightly better performance on this dataset.

3. **Poisson feedforward neural network model.**

Still working under the Poisson assumption, we can achieve a higher performance by modelling the mean parameter in (1) with a feedforward neural network, which can be used to perform feature engineering in an automatic way.

[2]For more information on feature engineering with fully worked-out examples for this dataset, see Section 5.3.4 of Wüthrich, Mario V., and Merz, Michael *Statistical foundations of actuarial learning and its applications.* Springer Nature, 2023.

(a) Implement a feedforward neural network $\hat{\lambda} = F_\theta(x)$ with exponential activation $x \mapsto \exp(x)$ in the output layer. You should experiment with different network architectures and hyperparameters, until you find a network that performs well. You can start with two hidden layers of 20 neurons each with ReLU activation function and train for 100 epochs with batch size 10,000 and learning rate 0.01.

(Bonus) Use a hyperparameter tuner (or *hypertuner*) to find optimal parameters for: `network depth, hidden nodes, inner activation function, learning rate`. See e.g. `https://www.tensorflow.org/tutorials/keras/keras_tuner`.

(b) Train the model on the training set of Exercise 1(a) by minimizing the loss $\mathcal{L}$ in Equation (2). You can implement the neural network using `Keras`. Remember to compile the model with the Poisson deviance as loss (you can use `keras.losses.Poisson`, but then justify why using this loss is equivalent to (2)) and to minimize the *weighted* Poisson deviance by fitting the model with the argument `sample_weight` (using `Exposure` as input). Report MAE, MSE and the loss $\mathcal{L}$ on train and test sets and show that the model outperforms the Poisson GLM from Exercise 1(b).

(Bonus) Try different regularization techniques, such as L^2-regularization. Remember to justify your choice of regularization parameter(s) using a grid-search procedure.