Data Analytics

Project Presentation

Claudia Citera - Riccardo Evangelisti

Academic Year 2023-2024

Introduction

Project purpose: to recognize the ***year*** in which a song was published based on the ***features of its audio track***.

To carry out a data analytics study we implemented all the **analytical pipeline** phases studied during the course

- Data Acquisition
- Data Visualization
- Data Preprocessing
- Modeling
- Performance Evaluation

Data Acquisition

Data Acquisition

Dataset Description

- It consists of a single csv file with over **250000 rows** and **91 columns**.
- One of the columns represents the **year** of publication of the song, ranging from 1956 to 2009. All other columns contain **floating-point numbers** related to the **audio track**.
- Since the error calculation for a classification problem would not distinguish the error gravity, **regression** models were used to solve the problem.

Data Acquisition

Data Cleaning

- Before performing any further actions, some data cleaning actions were performed.
- No **missing** values found
- Only 52 **duplicated** rows, which were from the most represented classes. We decided to **drop** them.

Data Acquisition

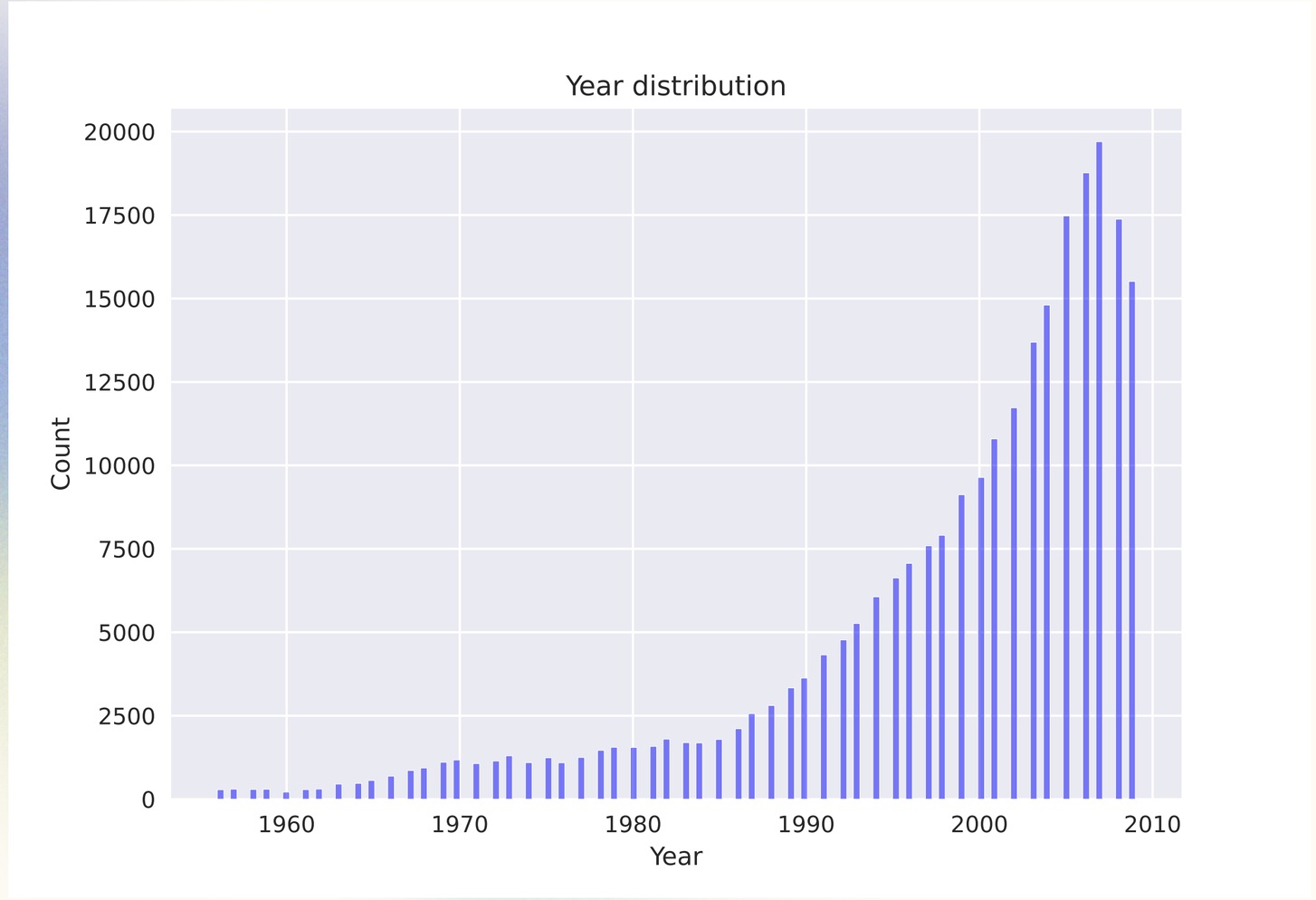
Dataset Management

- The dataset was splitted in 3 parts:
  - **Training** set (**70%**)
  - **Validation** set (**20%**)
  - **Test** set (**10%**)
- A seed was set to ensure the split's **reproducibility**
- The dataset is very **imbalanced**, so the "**stratify**" option (of the *train_test_split* function of *sklearn*) is set.

Data Visualisation

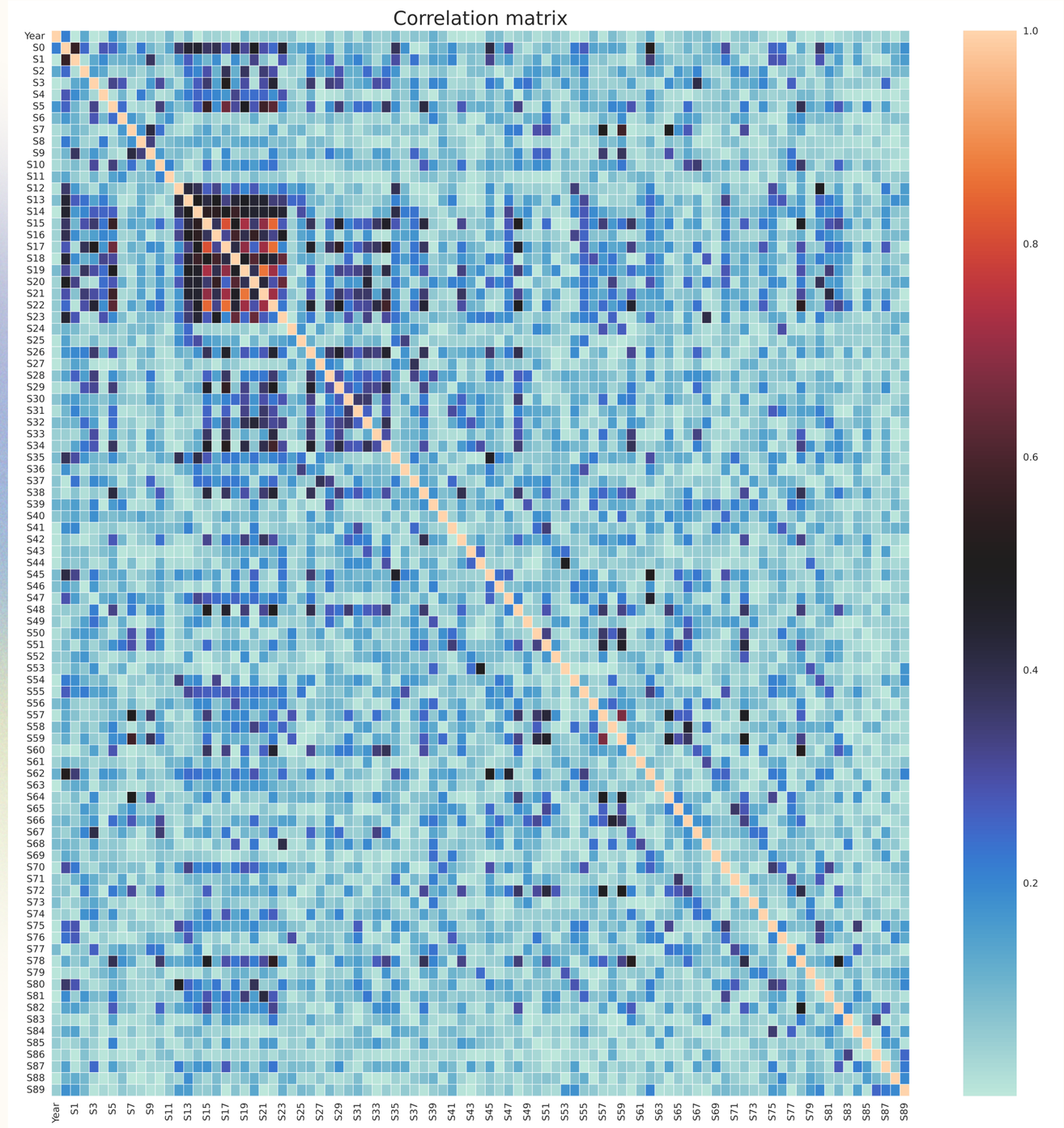
Data Visualisation

Year Distribution



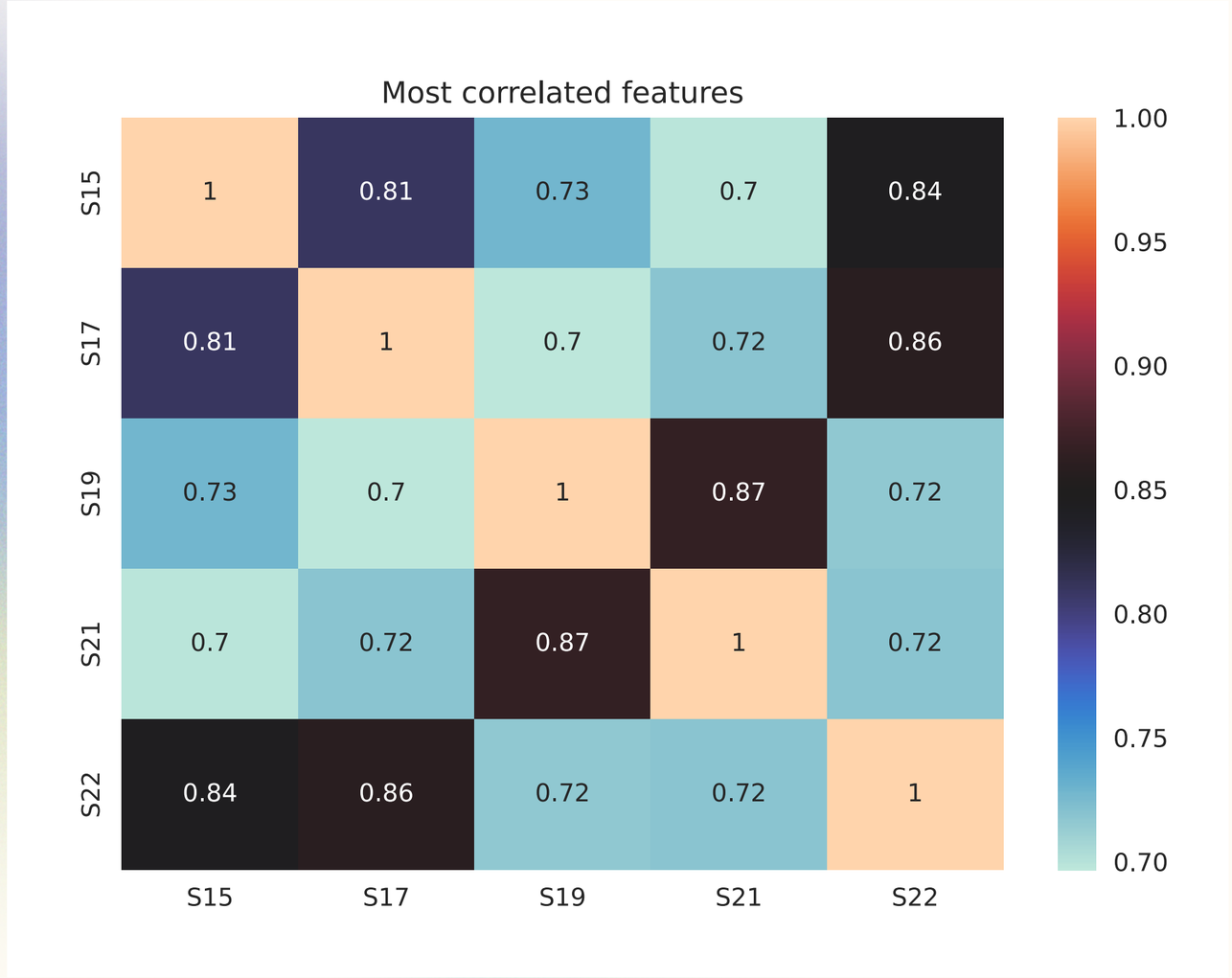
Data Visualisation

Correlation matrix between features



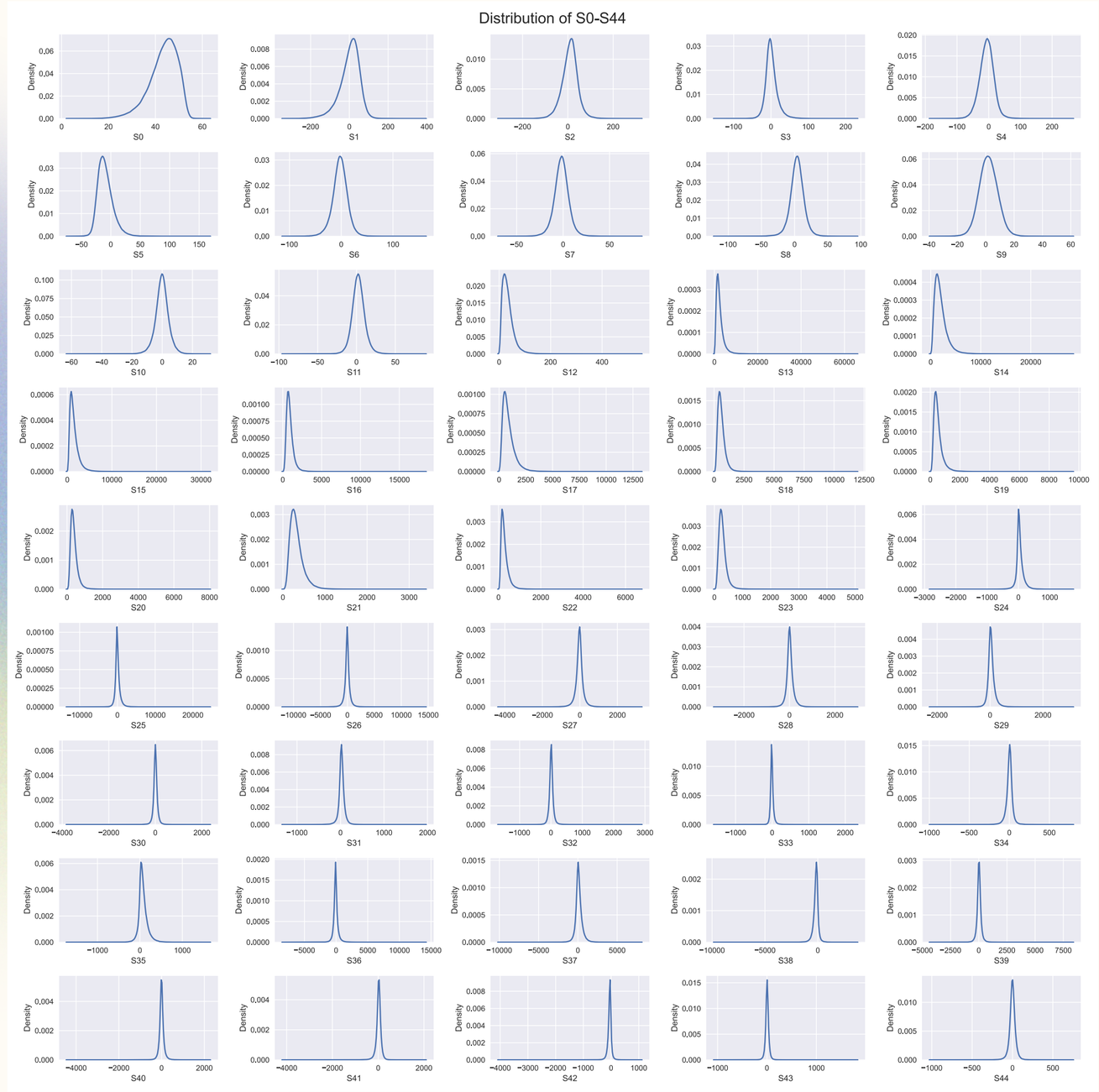
Data Visualisation

Correlation matrix close up



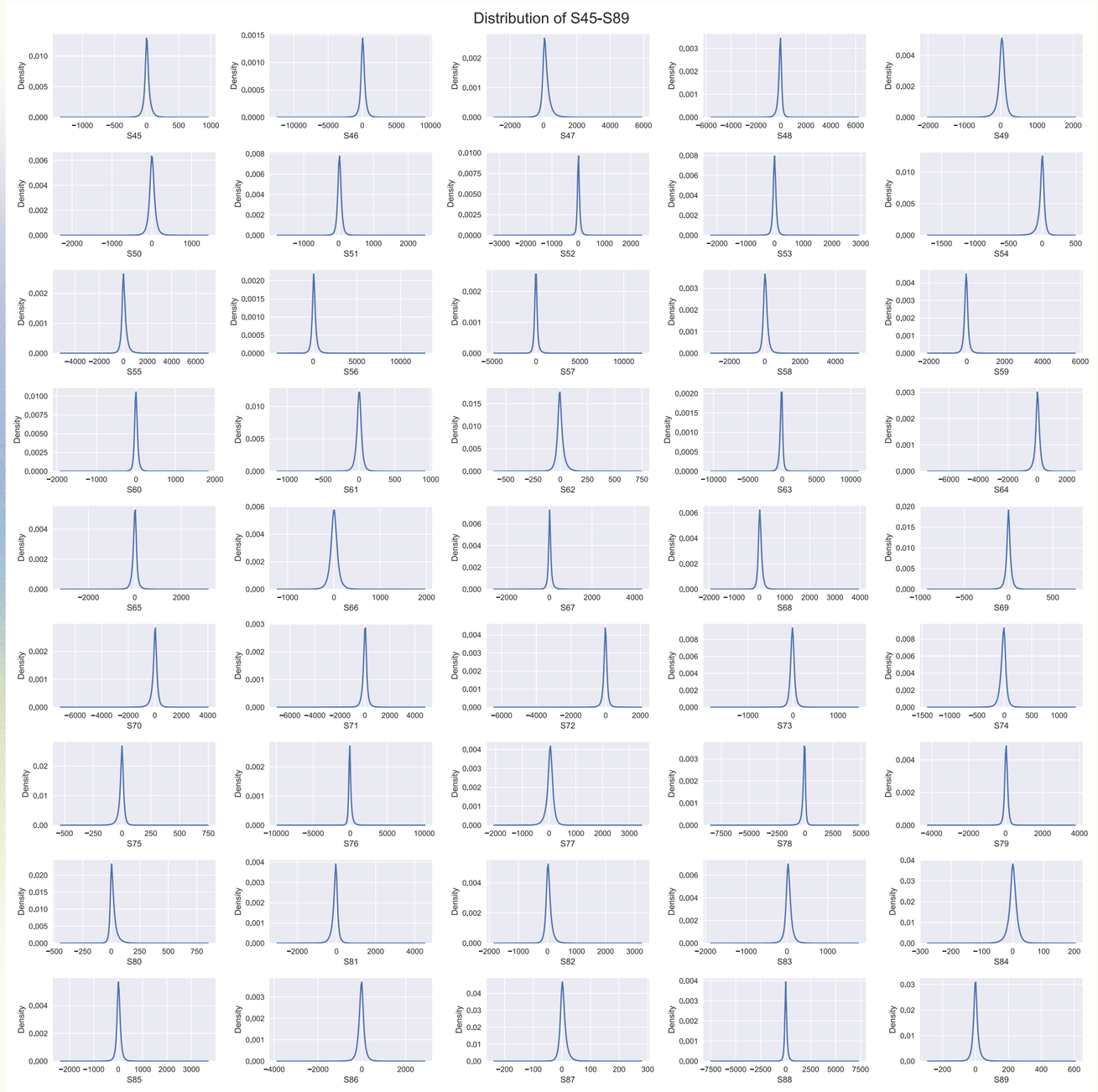
Data Visualisation

Distribution of S0-S44



Data Visualisation

Distribution of S45-S89



Data Preprocessing

Data Preprocessing

Various preprocessing techniques were applied and combined to find the best ones for each model.

- Scaling and Normalization
- Dimensionality Reduction
- Kernel Approximation
- Outlier Removal
- Under/Oversampling

Data Preprocessing

Scaling and Normalizing

- **Standard Scaling:** since some features don’t have the mean in zero
- **Min-Max Scaling:** since the features have different ranges
- **L1/L2/Lmax Normalization:** to enable fair and unbiased comparisons between vectors

Data Preprocessing

Dimensionality Reduction

- **PCA:** unsupervised method that uses an orthogonal transformation to convert a set of correlated variables into a set of uncorrelated variables → **54 Principal Components**
- **LDA:** supervised method used to project the data into a lower dimensional space that best separates the classes → **53 features**
  - Different Covariance estimators were tested

Data Preprocessing

Kernel Approximation

- Technique that approximates the feature mappings that correspond to certain kernel methods (like SVM)
- Since the dataset is very large, the **Nystroem Method** was tested with *RBF kernel* for the SVR task

Data Preprocessing

Outlier Removal

Outliers: data points that significantly differ from other observations.

Removing outliers can have a substantial impact on the results of machine learning models:

- **LocalOutlierFactor**
- **IsolationForest**

Data Preprocessing

Undersampling and Oversampling

To overcome the imbalance of the dataset, some Undersampling and Oversampling were tested:

- **RandomUnderSampling** on majority classes
- **SMOTE (Synthetic Minority Over-sampling)** on minority classes

Data Preprocessing - Best preprocessing pipeline per model

Model	Preprocessing Pipeline	Preprocessing parameter	
Linear Regressor	Min-Max + LDA + Nystroem	LDA cov. estimator Nys. gamma Nys. num. components	OAS 0.01 1000
Random Forest Regressor	Min-Max + LDA	LDA cov. estimator	OAS
K-Nearest Neighbors Regressor	Min-Max + Lmax + LDA	LDA cov. estimator	OAS
Support Vector Regressor	Min-Max + L1 + LDA + Nystroem	LDA cov. estimator Nys. gamma Nys. num. components	OAS 0.01 1000
Deep Neural Network	Standard Scaling + L2		
TabNet	Min-Max + Lmax + LDA	LDA cov. estimator	OAS
TabTransformer	Standard Scaling + L2		

Modeling

Modeling

Traditional non-deep supervised Machine Learning techniques

Modeling

Traditional ML

Linear Regression

Model that assumes a linear combination between the input features and the output.
It finds the best-fitting line that represents the relationship between the variables.

Modeling

Traditional ML

Random Forest Regressor

Algorithm based on creating an ensemble of decision trees, where each tree is built on a random subset of the training data, resulting in different models, and uses only a random subset of input variables and attributes to make predictions.

Modeling

Traditional ML

Random Forest Regressor

Hyperparameters	Values
max_samples	0.1, 0.5, 0.66, 0.8, 1.0
n_estimators	5, 100, 200, 500
max_depth	5, 10, 15, 20, 25, None
max_leaf_nodes	50, 100, 200, 300, 400, None
ccp_alpha	0.0, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0
max_features	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, "sqrt", "log2", None

Modeling

Traditional ML

K-Nearest Neighbors Regressor

Regression model based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

Hyperparameters	Values
n_neighbors	5, 10, 15, 19, 20, 21, 22, 25
weights	"distance", "uniform"
metric	"cosine", "euclidean", "cityblock", "nan_euclidean"

Modeling

Traditional ML

Support Vector Regressor

SVR is a machine learning method that aims to find a hyperplane that maximally separates the support vectors, ensuring the maximum distance between the hyperplane and the support vectors.

For computaional reasons, we opted for the linear kernel version of the algorithm: **LinearSVR**

Modeling

Traditional ML

Support Vector Regressor

Hyperparameters	Values
epsilon	0, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0
C	0.1, 1, 5, 8, 9, 10, 15, 20, 100, 500
max_iter	2000
tol	1e-5, 1e-4, 1e-3, 1e-2

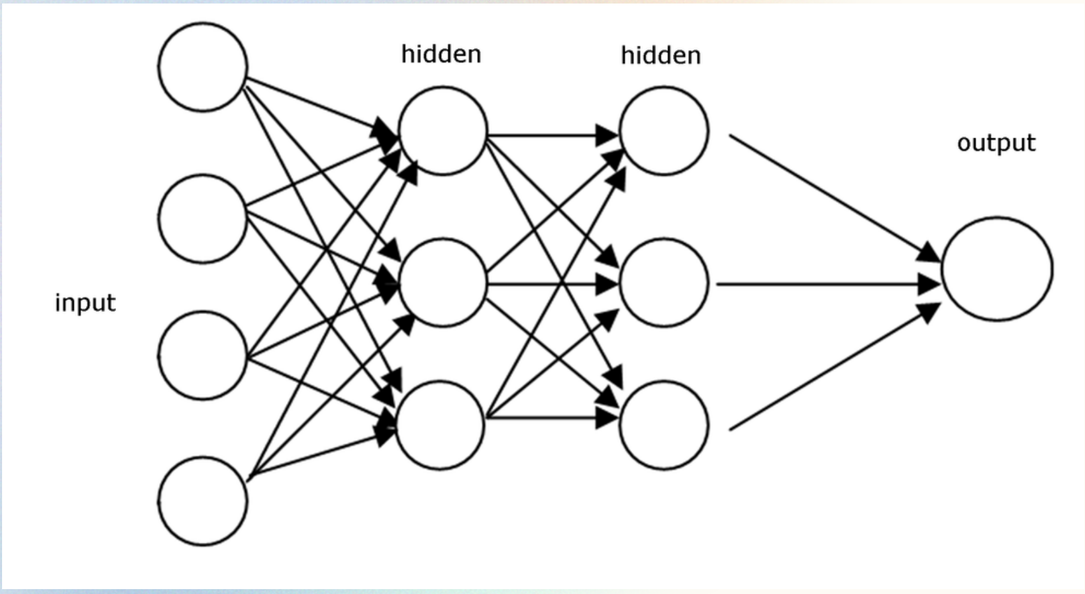
Modeling

Supervised ML techniques based on Neural Networks

Modeling

Neural Networks

- The neural network was implemented using the **PyTorch** framework.
- The network used is a **FeedForward** model with a variable number of layers.
- Several architectures have been tested.



Modeling

Neural Networks

DNN hyperparameters

- **epochs:** the number of training iterations → [30, 100, 200]
- **hidden_size:** the size (number of neurons) in the hidden layers →[32, 64, 128, 256, 512, 1024]
- **batch_size:** how many samples per batch → [8, 16, 32, 48, 64]
- **depth:** the number of hidden layers in the neural network → [2, 3, 4, 5]
- **dropout_rate:** the probability that some neurons are deactivated during network training → [0.1, 0.2, 0.3, 0.5]

Modeling

Neural Networks

DNN hyperparameters

- **criterion:** the loss function → [nn.MSELoss(), nn.L1Loss(), nn.HuberLoss()]
- **early_stopper_patience:** the number of epochs before stopping the training if the validation loss stops improving → [30]
- **optimizer:** the network optimizer → [RMSprop, Adam]
- **momentum:** RMSprop momentum → [0.9]
- **learning_rate**: the step size during gradient descent optimization → [0.0001, 0.001, 0.05, 0.1]

Modeling

Neural Networks

Scheduler parameters

- **lr_scheduler** → [ReduceLROnPlateau]
- **patience:** number of epochs with no improvement after which learning rate will be reduced → [10, 15, 20, 21, 30, 200]
- **threshold:** threshold for measuring the new optimum, to only focus on significant changes → [1]
- **threshold_mode** → [rel, abs]
- **factor:** factor by which the learning rate will be reduced → [0.1]
- **min_lr:** the lower bound on the learning rate → [0.0000001]

Modeling

Supervised ML technique with deep models for Tabular Data

Modeling

Tabular Data

PyTorch Tabular

- **PyTorch Tabular** is a powerful library that aims to simplify and popularize the application of deep learning techniques to tabular data.
- The models of this module are capable of handling large amounts of data and finding complex relationships between variables, improving the model’s predictive ability.
- **TabNet** and **TabTransformer** are two different models for tabular data processing in this library.

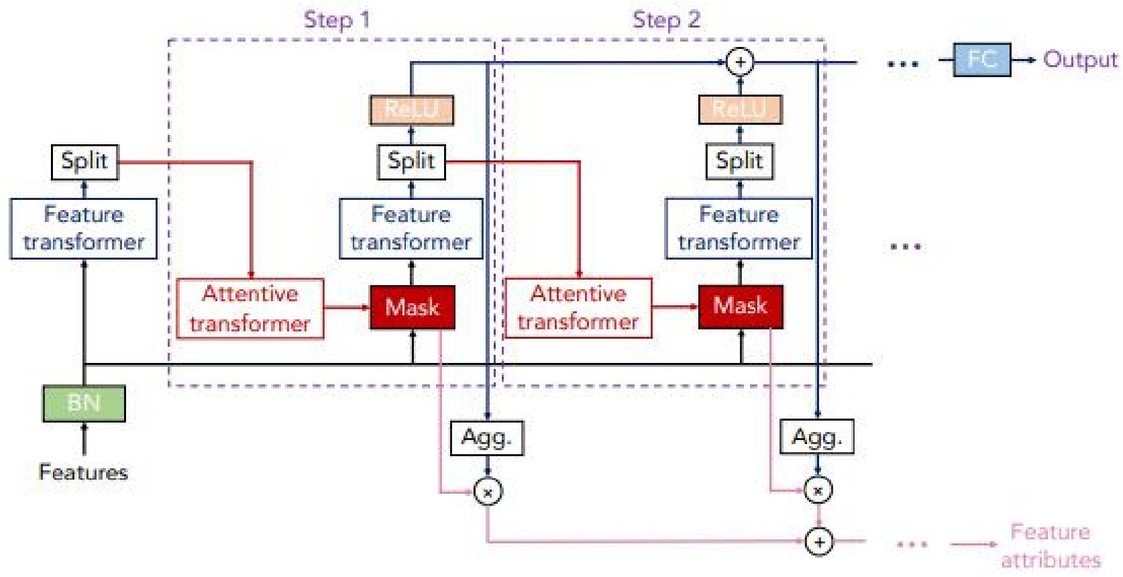
Modeling

Tabular Data

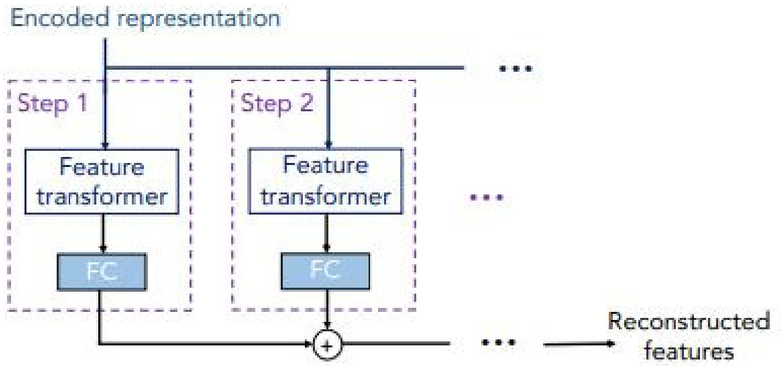
TabNet

- **TabNet** is a deep learning algorithm based on **sequential attention**.
- The model consists of **several cascading neural networks.**
- It’s based on 3 principal elements:
  - **Encoder**
  - **Mask**
  - **Decoder**

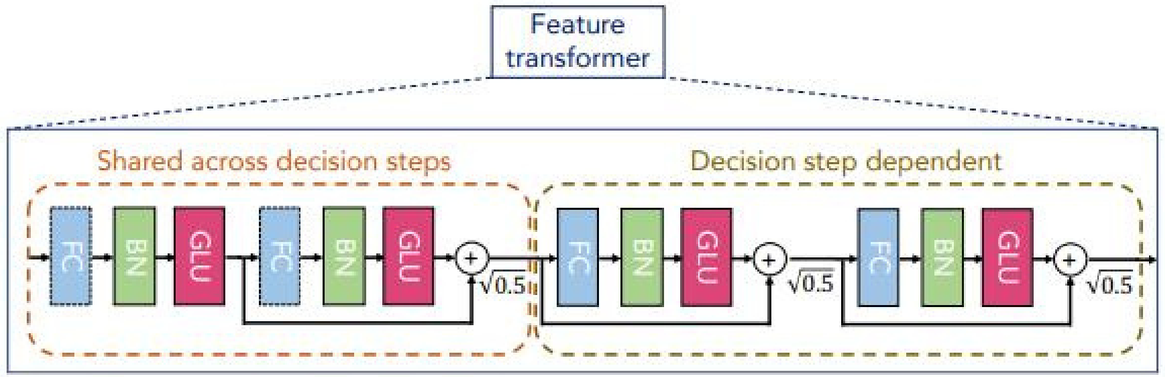
TabNet - Architecture



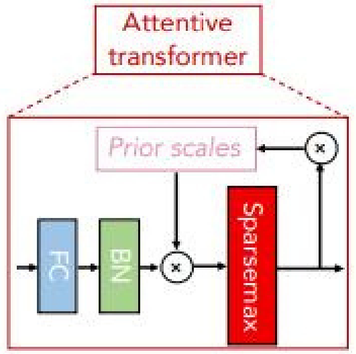
(a) TabNet encoder architecture



(b) TabNet decoder architecture



(c)



(d)

Modeling

Tabular Data - TabNet

***Hyperparameters* (1)**

- **max_epochs:** maximum number of epochs to be run → [200]
- **early_stopping_mode** → [min]
- **early_stopping_patience** → [10]
- **batch_size:** number of samples in each batch of training → [256, 512]
- **optimizer:** the network optimizer → [Adam]
- **lr_scheduler** → [ReduceLROnPlateau]
- **lr_scheduler_params**
  - patience: [7, 8, 9, 10, 11, 12]
  - threshold: [1]
  - threshold_mode: [abs]

Modeling

Tabular Data - TabNet

***Hyperparameters* (2)**

- **learning_rate:** the learning rate of the model →[0.005]
- **virtual_batch_size:** batch size for Ghost Batch Normalization → [128]
- **n_independent:** the number of independent features to use in the transformer network → [2, 3]
- **n_shared:** the number of shared features to use in the transformer network → [2, 3]
- **n_a:** dimension of the attention layer → [16, 32]
- **n_d:** dimension of the prediction layer → [32, 64]
- **n_step:** number of successive steps in the network → [3]
- **gamma:** scaling factor for attention updates → [1.5]
- **seed:** the random seed to use for reproducibility → [42]

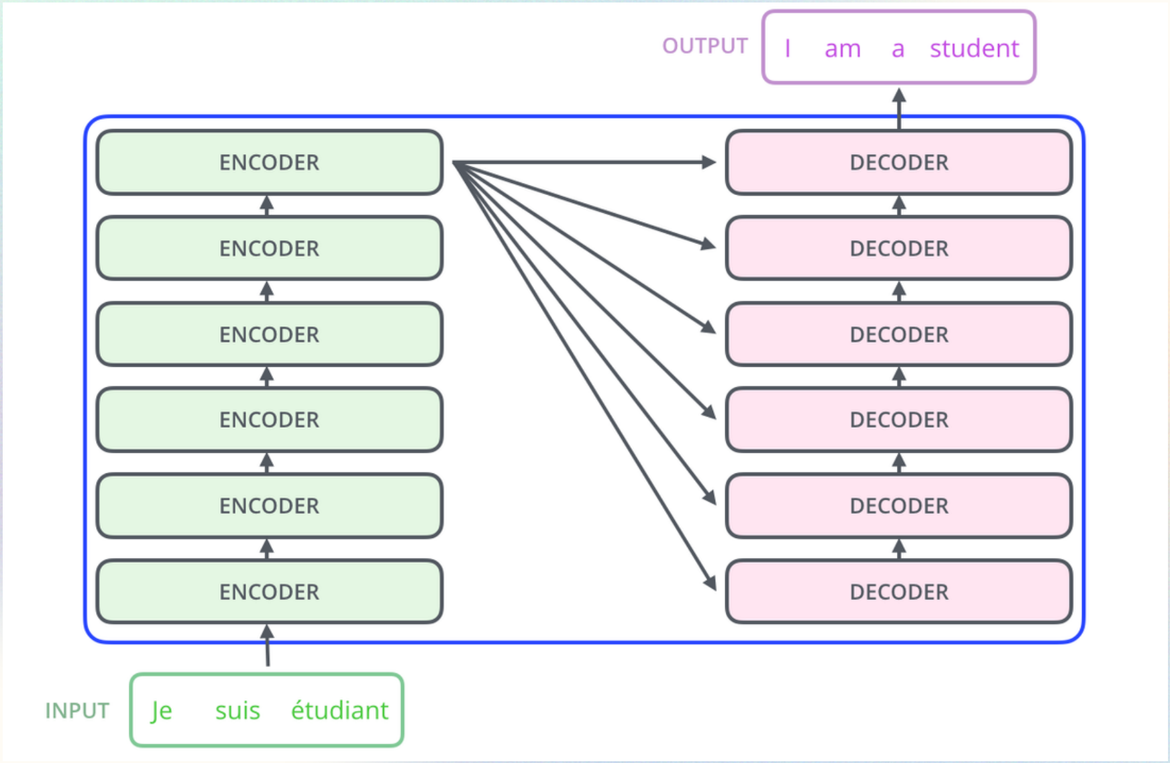
Modeling

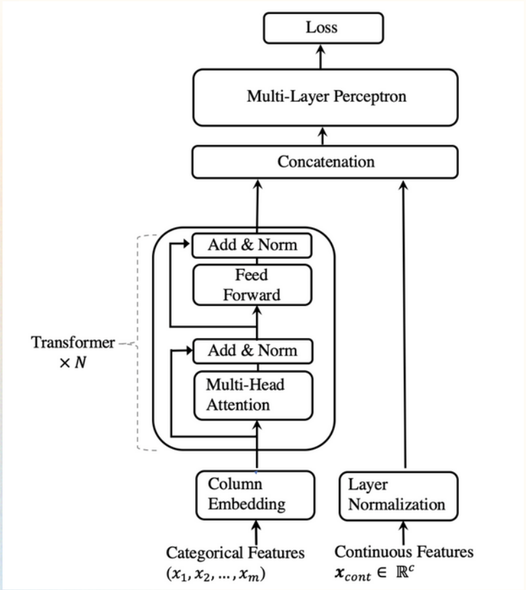
Tabular Data

TabTransformer

- TabTransformer is a **transformer-based** model that uses **multi-head attention** to capture the dependencies between features and is known for its ability to handle tabular data.
- It leverages transformers to transform categorical feature **embeddings**.

TabTransformer





Modeling

Tabular Data - TabTransformer

***Hyperparameters* (1)**

- **max_epochs:** maximum number of epochs to be run → [100]
- **batch_size:** number of samples in each batch of training → [512, 1024]
- **optimizer**: the network optimizer → [Adam]
- **lr_scheduler** → [ReduceLROnPlateau]
- **lr_scheduler_params**
  - patience: [7, 8, 9, 10, 11, 12]
  - threshold: [1]
  - threshold_mode: [abs]

Modeling

Tabular Data - TabTransformer

***Hyperparameters* (2)**

- **learning_rate:** the learning rate of the model → [0.01]
- **virtual_batch_size:** batch size for Ghost Batch Normalization → [64, 128]
- **num_heads:** the number of heads in the Multi-Headed Attention layer → [8]
- **num_attn_blocks:** the number of layers of stacked Multi-Headed Attention layers → [6]
- **transform_activation:** the activation type in the transformer feed-forward layers → [’ReGLU’, ’GEGLU’, ’SwiGLU’]
- **seed:** the random seed to use for reproducibility → [42]

Performance Evaluation

Performance Evaluation

Linear Regression

Preprocessing

Min-Max + LDA + Nystroem

Results

MSE: 73.598

MAE: 6.123

MAPE: 0.0031

R2 score: 0.331

Performance Evaluation

Random Forest Regressor

Preprocessing

Min-Max + LDA

Best Hyperparameters

Hyperparameter	Value
max_samples	0.66
n_estimators	200
max_depth	25
max_leaf_nodes	None
ccp_alpha	0.01
max_features	0.4

Performance Evaluation

Random Forest Regressor

Preprocessing

Min-Max + LDA

Results

MSE: 73.903

MAE: 6.059

MAPE: 0.0030

R2 score: 0.329

Performance Evaluation

KNN Regressor

Preprocessing

Min-Max + Lmax + LDA

Best Hyperparameters

Hyperparameter	Value
n_neighbors	21
weights	“distance”
metric	cosine

Performance Evaluation

KNN Regressor

Preprocessing

Min-Max + Lmax + LDA

Results

MSE: 73.656

MAE: 6.095

MAPE: 0.0031

R2 score: 0.331

Performance Evaluation

Support Vector Regressor

Preprocessing

Min-Max + L1 + LDA + Nystroem

Best Hyperparameters

Hyperparameter	Value
epsilon	0.01
C	10
max_iter	2000
tol	0.0001

Performance Evaluation

Support Vector Regressor

Preprocessing

Min-Max + L1 + LDA + Nystroem

Results

MSE: 78.721

MAE: 5.880

MAPE: 0.0030

R2 score: 0.285

Performance Evaluation

Neural Network

Preprocessing

Standard Scaling + L2

Best Hyperparameters

Hyperparameter	Value
epochs	30
hidden_size	256
batch_size	48
depth	2
learning_rate	0.0001
criterion	MSELoss()

Performance Evaluation

Neural Network

Preprocessing

Standard Scaling + L2

Results

MSE: 70.935

MAE: 5.978

MAPE: 0.0030

R2 score: 0.356

Performance Evaluation

TabNet

Preprocessing

Min-Max + Lmax + LDA

Best Hyperparameters

Hyperparameter	Value	Hyperparameter	Value
learning_rate	0.005	n_a	16
batch_size	256	n_step	3
virtual_batch_size	128	n_independent	2
n_epochs	200	n_shared	2
n_d	64	gamma	1.5

Performance Evaluation

TabNet

Preprocessing

Min-Max + Lmax + LDA

Results

MSE: 70.756

MAE: 5.934

MAPE: 0.0030

R2 score: 0.357

Performance Evaluation

TabTransformer

Preprocessing

Standard Scaling + L2

Best Hyperparameters

Hyperparameter	Value
learning_rate	0.01
batch_size	512
virtual_batch_size	128
n_epochs	100

Performance Evaluation

TabTransformer

Preprocessing

Standard Scaling + L2

Results

MSE: 78.044

MAE: 6.329

MAPE: 0.0032

R2 score: 0.291

Performance Evaluation

Results Recap

Model	MSE	MAE	MAPE	R2
LR	73.598	6.123	0.0031	0.331
RF	73.903	6.059	0.0030	0.329
KNR	73.656	6.095	0.0031	0.331
SVR	78.721	5.880	0.0030	0.285
FF	70.935	5.978	0.0030	0.356
TB	70.756	5.934	0.0030	0.357
TF	78.044	6.329	0.0032	0.291

Thank you!