TabNet_Interpretability_out

July 27, 2022

1 TabNet Interpretability

TabNet is designed to learn a 'decision-tree-like' mapping in order to inherit the valuable benefits of tree-based methods (explainability) while providing the key benefits of deep learning-based methods (high performance & new capabilities)

Here we will look at the interpretability of TabNet using the Friedman3 dataset.

1.1 Importing Libraries

[1]:
```
import warnings
import sys
sys.path.insert(0, '../../src')
warnings.filterwarnings('ignore')

import numpy  as np
import pandas as pd

# plotting
import matplotlib.pyplot as plt
import plotly.express as px

from pytorch_tabnet.tab_model import TabNetRegressor
import torch
import friedman3Dataset
import dataset
from sklearn.model_selection import train_test_split
```

1.2 Importing the dataset

[2]:
```
n_features = 4
n_samples= 100
n_target = 1
X,Y = friedman3Dataset.friedman3_data(n_samples)

# Train test split for dataset
real_dataset = dataset.CustomDataset(X,Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

```
y_train = np.reshape(y_train, (-1, 1))
y_test = np.reshape(y_test, (-1, 1))
```

	X1	X2	X3	X4	Y
0	55.651865	1654.311793	0.208636	7.307353	1.464615
1	88.063689	1371.168212	0.935396	8.498412	1.507733
2	96.584120	491.982521	0.096732	1.906904	0.575998
3	20.511168	562.839797	0.072542	4.082258	1.088568
4	35.749655	1346.316492	0.407447	1.689462	1.485549

1.3 TabNet Regressor

[3]:
```
n_epochs =1000
batch_size = 32

regressor = TabNetRegressor(optimizer_fn=torch.optim.Adam,
                        optimizer_params=dict(lr = 0.001),
                        mask_type= 'sparsemax',
                        verbose = 1)

regressor.fit(X_train = X_train,y_train = y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        eval_name=['train', 'valid'],
        eval_metric=[ 'mae'],
        max_epochs = n_epochs,
        batch_size = batch_size,
        patience=50)
```

```
Device used : cpu
epoch 0  | loss: 1.29373 | train_mae: 1.56357 | valid_mae: 1.80989 |  0:00:00s
epoch 1  | loss: 1.39795 | train_mae: 1.453   | valid_mae: 1.51107 |  0:00:00s
epoch 2  | loss: 1.34452 | train_mae: 1.35179 | valid_mae: 1.4094  |  0:00:00s
epoch 3  | loss: 1.03391 | train_mae: 1.25303 | valid_mae: 1.28679 |  0:00:01s
epoch 4  | loss: 0.91463 | train_mae: 1.19144 | valid_mae: 1.16838 |  0:00:01s
epoch 5  | loss: 0.98312 | train_mae: 1.09752 | valid_mae: 1.05087 |  0:00:01s
epoch 6  | loss: 0.91217 | train_mae: 1.01259 | valid_mae: 1.01749 |  0:00:01s
epoch 7  | loss: 0.85831 | train_mae: 0.97565 | valid_mae: 0.96591 |  0:00:01s
epoch 8  | loss: 0.79473 | train_mae: 0.93478 | valid_mae: 0.90319 |  0:00:01s
epoch 9  | loss: 0.75725 | train_mae: 0.89824 | valid_mae: 0.87434 |  0:00:01s
epoch 10 | loss: 0.82541 | train_mae: 0.86612 | valid_mae: 0.83926 |  0:00:02s
epoch 11 | loss: 0.75563 | train_mae: 0.82665 | valid_mae: 0.8179  |  0:00:02s
epoch 12 | loss: 0.74831 | train_mae: 0.79534 | valid_mae: 0.79311 |  0:00:02s
epoch 13 | loss: 0.71694 | train_mae: 0.76027 | valid_mae: 0.76543 |  0:00:02s
epoch 14 | loss: 0.7164  | train_mae: 0.73868 | valid_mae: 0.7395  |  0:00:02s
epoch 15 | loss: 0.70595 | train_mae: 0.71518 | valid_mae: 0.71467 |  0:00:02s
epoch 16 | loss: 0.69947 | train_mae: 0.69377 | valid_mae: 0.69188 |  0:00:02s
```

```
epoch 17 | loss: 0.67541 | train_mae: 0.67334 | valid_mae: 0.67308 |  0:00:02s
epoch 18 | loss: 0.78273 | train_mae: 0.65801 | valid_mae: 0.6678  |  0:00:02s
epoch 19 | loss: 0.68276 | train_mae: 0.63728 | valid_mae: 0.66362 |  0:00:03s
epoch 20 | loss: 0.70407 | train_mae: 0.61348 | valid_mae: 0.66225 |  0:00:03s
epoch 21 | loss: 0.63186 | train_mae: 0.6034  | valid_mae: 0.65817 |  0:00:03s
epoch 22 | loss: 0.58956 | train_mae: 0.5906  | valid_mae: 0.6422  |  0:00:03s
epoch 23 | loss: 0.65525 | train_mae: 0.57727 | valid_mae: 0.62997 |  0:00:03s
epoch 24 | loss: 0.52221 | train_mae: 0.56528 | valid_mae: 0.61163 |  0:00:03s
epoch 25 | loss: 0.64885 | train_mae: 0.54755 | valid_mae: 0.60122 |  0:00:03s
epoch 26 | loss: 0.63516 | train_mae: 0.53014 | valid_mae: 0.59548 |  0:00:03s
epoch 27 | loss: 0.55926 | train_mae: 0.51895 | valid_mae: 0.59048 |  0:00:03s
epoch 28 | loss: 0.5031  | train_mae: 0.50383 | valid_mae: 0.57681 |  0:00:03s
epoch 29 | loss: 0.57318 | train_mae: 0.49266 | valid_mae: 0.56479 |  0:00:03s
epoch 30 | loss: 0.44983 | train_mae: 0.48052 | valid_mae: 0.55973 |  0:00:04s
epoch 31 | loss: 0.43049 | train_mae: 0.46995 | valid_mae: 0.5507  |  0:00:04s
epoch 32 | loss: 0.446   | train_mae: 0.45882 | valid_mae: 0.54815 |  0:00:04s
epoch 33 | loss: 0.52657 | train_mae: 0.44478 | valid_mae: 0.54226 |  0:00:04s
epoch 34 | loss: 0.48354 | train_mae: 0.43332 | valid_mae: 0.53023 |  0:00:04s
epoch 35 | loss: 0.32463 | train_mae: 0.42722 | valid_mae: 0.52392 |  0:00:04s
epoch 36 | loss: 0.55819 | train_mae: 0.41721 | valid_mae: 0.51684 |  0:00:04s
epoch 37 | loss: 0.42494 | train_mae: 0.40471 | valid_mae: 0.50444 |  0:00:04s
epoch 38 | loss: 0.47651 | train_mae: 0.39293 | valid_mae: 0.49626 |  0:00:04s
epoch 39 | loss: 0.36501 | train_mae: 0.38283 | valid_mae: 0.48486 |  0:00:04s
epoch 40 | loss: 0.25255 | train_mae: 0.37471 | valid_mae: 0.47397 |  0:00:05s
epoch 41 | loss: 0.45012 | train_mae: 0.37108 | valid_mae: 0.47291 |  0:00:05s
epoch 42 | loss: 0.40512 | train_mae: 0.37294 | valid_mae: 0.47482 |  0:00:05s
epoch 43 | loss: 0.40635 | train_mae: 0.3698  | valid_mae: 0.46685 |  0:00:05s
epoch 44 | loss: 0.34338 | train_mae: 0.36547 | valid_mae: 0.46491 |  0:00:05s
epoch 45 | loss: 0.30325 | train_mae: 0.36435 | valid_mae: 0.4691  |  0:00:05s
epoch 46 | loss: 0.25797 | train_mae: 0.36119 | valid_mae: 0.46715 |  0:00:05s
epoch 47 | loss: 0.30518 | train_mae: 0.35587 | valid_mae: 0.45659 |  0:00:05s
epoch 48 | loss: 0.26272 | train_mae: 0.35356 | valid_mae: 0.45385 |  0:00:05s
epoch 49 | loss: 0.38817 | train_mae: 0.35202 | valid_mae: 0.45416 |  0:00:05s
epoch 50 | loss: 0.24284 | train_mae: 0.34988 | valid_mae: 0.45586 |  0:00:06s
epoch 51 | loss: 0.21269 | train_mae: 0.34458 | valid_mae: 0.45456 |  0:00:06s
epoch 52 | loss: 0.38177 | train_mae: 0.3396  | valid_mae: 0.45935 |  0:00:06s
epoch 53 | loss: 0.2302  | train_mae: 0.33711 | valid_mae: 0.46228 |  0:00:06s
epoch 54 | loss: 0.32454 | train_mae: 0.33586 | valid_mae: 0.45781 |  0:00:06s
epoch 55 | loss: 0.33505 | train_mae: 0.33256 | valid_mae: 0.45211 |  0:00:06s
epoch 56 | loss: 0.19674 | train_mae: 0.32633 | valid_mae: 0.44098 |  0:00:06s
epoch 57 | loss: 0.32704 | train_mae: 0.32377 | valid_mae: 0.43675 |  0:00:06s
epoch 58 | loss: 0.30353 | train_mae: 0.32351 | valid_mae: 0.43479 |  0:00:06s
epoch 59 | loss: 0.23147 | train_mae: 0.31993 | valid_mae: 0.43274 |  0:00:06s
epoch 60 | loss: 0.38049 | train_mae: 0.32074 | valid_mae: 0.43828 |  0:00:07s
epoch 61 | loss: 0.2703  | train_mae: 0.32071 | valid_mae: 0.44221 |  0:00:07s
epoch 62 | loss: 0.41121 | train_mae: 0.32176 | valid_mae: 0.44248 |  0:00:07s
epoch 63 | loss: 0.17391 | train_mae: 0.32316 | valid_mae: 0.44041 |  0:00:07s
epoch 64 | loss: 0.18706 | train_mae: 0.32198 | valid_mae: 0.43934 |  0:00:07s
```

```
epoch 65 | loss: 0.28812 | train_mae: 0.32139 | valid_mae: 0.4395  |  0:00:07s
epoch 66 | loss: 0.24915 | train_mae: 0.32075 | valid_mae: 0.43721 |  0:00:07s
epoch 67 | loss: 0.39802 | train_mae: 0.31821 | valid_mae: 0.43124 |  0:00:07s
epoch 68 | loss: 0.19791 | train_mae: 0.31187 | valid_mae: 0.43096 |  0:00:07s
epoch 69 | loss: 0.19912 | train_mae: 0.30469 | valid_mae: 0.42283 |  0:00:07s
epoch 70 | loss: 0.25662 | train_mae: 0.29837 | valid_mae: 0.41137 |  0:00:07s
epoch 71 | loss: 0.31487 | train_mae: 0.295   | valid_mae: 0.3981  |  0:00:08s
epoch 72 | loss: 0.18967 | train_mae: 0.2939  | valid_mae: 0.38906 |  0:00:08s
epoch 73 | loss: 0.20114 | train_mae: 0.29229 | valid_mae: 0.38297 |  0:00:08s
epoch 74 | loss: 0.23036 | train_mae: 0.29319 | valid_mae: 0.37578 |  0:00:08s
epoch 75 | loss: 0.22982 | train_mae: 0.28927 | valid_mae: 0.37587 |  0:00:08s
epoch 76 | loss: 0.30488 | train_mae: 0.28485 | valid_mae: 0.37975 |  0:00:08s
epoch 77 | loss: 0.39686 | train_mae: 0.2841  | valid_mae: 0.37919 |  0:00:08s
epoch 78 | loss: 0.28249 | train_mae: 0.28406 | valid_mae: 0.37704 |  0:00:08s
epoch 79 | loss: 0.20412 | train_mae: 0.2826  | valid_mae: 0.37613 |  0:00:08s
epoch 80 | loss: 0.19851 | train_mae: 0.2806  | valid_mae: 0.37031 |  0:00:08s
epoch 81 | loss: 0.18562 | train_mae: 0.27777 | valid_mae: 0.3699  |  0:00:09s
epoch 82 | loss: 0.1938  | train_mae: 0.27424 | valid_mae: 0.37172 |  0:00:09s
epoch 83 | loss: 0.166   | train_mae: 0.27036 | valid_mae: 0.36943 |  0:00:09s
epoch 84 | loss: 0.18277 | train_mae: 0.26776 | valid_mae: 0.37074 |  0:00:09s
epoch 85 | loss: 0.26758 | train_mae: 0.27033 | valid_mae: 0.37284 |  0:00:09s
epoch 86 | loss: 0.1868  | train_mae: 0.2729  | valid_mae: 0.37562 |  0:00:09s
epoch 87 | loss: 0.25274 | train_mae: 0.2758  | valid_mae: 0.37738 |  0:00:09s
epoch 88 | loss: 0.18719 | train_mae: 0.27605 | valid_mae: 0.38064 |  0:00:09s
epoch 89 | loss: 0.21438 | train_mae: 0.27544 | valid_mae: 0.37934 |  0:00:09s
epoch 90 | loss: 0.27615 | train_mae: 0.27209 | valid_mae: 0.37895 |  0:00:09s
epoch 91 | loss: 0.21061 | train_mae: 0.27142 | valid_mae: 0.37407 |  0:00:09s
epoch 92 | loss: 0.15168 | train_mae: 0.27373 | valid_mae: 0.36932 |  0:00:10s
epoch 93 | loss: 0.1717  | train_mae: 0.27449 | valid_mae: 0.37242 |  0:00:10s
epoch 94 | loss: 0.20967 | train_mae: 0.27445 | valid_mae: 0.37234 |  0:00:10s
epoch 95 | loss: 0.14806 | train_mae: 0.27542 | valid_mae: 0.37752 |  0:00:10s
epoch 96 | loss: 0.16372 | train_mae: 0.27531 | valid_mae: 0.38142 |  0:00:10s
epoch 97 | loss: 0.30797 | train_mae: 0.27737 | valid_mae: 0.3864  |  0:00:10s
epoch 98 | loss: 0.16066 | train_mae: 0.27592 | valid_mae: 0.38816 |  0:00:10s
epoch 99 | loss: 0.17942 | train_mae: 0.2798  | valid_mae: 0.39114 |  0:00:10s
epoch 100| loss: 0.5489  | train_mae: 0.28255 | valid_mae: 0.39521 |  0:00:10s
epoch 101| loss: 0.23339 | train_mae: 0.28269 | valid_mae: 0.39499 |  0:00:10s
epoch 102| loss: 0.26487 | train_mae: 0.28514 | valid_mae: 0.40267 |  0:00:10s
epoch 103| loss: 0.16882 | train_mae: 0.28576 | valid_mae: 0.40037 |  0:00:11s
epoch 104| loss: 0.30467 | train_mae: 0.28379 | valid_mae: 0.40238 |  0:00:11s
epoch 105| loss: 0.1252  | train_mae: 0.28307 | valid_mae: 0.40177 |  0:00:11s
epoch 106| loss: 0.21787 | train_mae: 0.27865 | valid_mae: 0.397   |  0:00:11s
epoch 107| loss: 0.26954 | train_mae: 0.27892 | valid_mae: 0.39858 |  0:00:11s
epoch 108| loss: 0.26731 | train_mae: 0.27406 | valid_mae: 0.40033 |  0:00:11s
epoch 109| loss: 0.17022 | train_mae: 0.27266 | valid_mae: 0.40208 |  0:00:11s
epoch 110| loss: 0.21207 | train_mae: 0.2733  | valid_mae: 0.40023 |  0:00:11s
epoch 111| loss: 0.21274 | train_mae: 0.2744  | valid_mae: 0.3975  |  0:00:11s
epoch 112| loss: 0.28419 | train_mae: 0.27269 | valid_mae: 0.39699 |  0:00:11s
```

```
epoch 113| loss: 0.19954 | train_mae: 0.2669  | valid_mae: 0.39209 |  0:00:11s
epoch 114| loss: 0.19703 | train_mae: 0.26447 | valid_mae: 0.39214 |  0:00:12s
epoch 115| loss: 0.21858 | train_mae: 0.26711 | valid_mae: 0.38065 |  0:00:12s
epoch 116| loss: 0.21822 | train_mae: 0.26304 | valid_mae: 0.37432 |  0:00:12s
epoch 117| loss: 0.29028 | train_mae: 0.25901 | valid_mae: 0.3674  |  0:00:12s
epoch 118| loss: 0.19244 | train_mae: 0.25586 | valid_mae: 0.37289 |  0:00:12s
epoch 119| loss: 0.14118 | train_mae: 0.25803 | valid_mae: 0.3739  |  0:00:12s
epoch 120| loss: 0.28781 | train_mae: 0.26171 | valid_mae: 0.37642 |  0:00:12s
epoch 121| loss: 0.2238  | train_mae: 0.26048 | valid_mae: 0.37278 |  0:00:12s
epoch 122| loss: 0.1917  | train_mae: 0.26439 | valid_mae: 0.37263 |  0:00:12s
epoch 123| loss: 0.25387 | train_mae: 0.2515  | valid_mae: 0.37021 |  0:00:12s
epoch 124| loss: 0.14585 | train_mae: 0.25741 | valid_mae: 0.37183 |  0:00:13s
epoch 125| loss: 0.28887 | train_mae: 0.26009 | valid_mae: 0.37338 |  0:00:13s
epoch 126| loss: 0.26721 | train_mae: 0.26379 | valid_mae: 0.37222 |  0:00:13s
epoch 127| loss: 0.21319 | train_mae: 0.27037 | valid_mae: 0.37093 |  0:00:13s
epoch 128| loss: 0.33748 | train_mae: 0.27023 | valid_mae: 0.36999 |  0:00:13s
epoch 129| loss: 0.16745 | train_mae: 0.26845 | valid_mae: 0.36936 |  0:00:13s
epoch 130| loss: 0.17692 | train_mae: 0.27043 | valid_mae: 0.37203 |  0:00:13s
epoch 131| loss: 0.25555 | train_mae: 0.26747 | valid_mae: 0.37194 |  0:00:13s
epoch 132| loss: 0.18991 | train_mae: 0.27114 | valid_mae: 0.37546 |  0:00:13s
epoch 133| loss: 0.34972 | train_mae: 0.27368 | valid_mae: 0.37609 |  0:00:13s
epoch 134| loss: 0.29443 | train_mae: 0.27456 | valid_mae: 0.37416 |  0:00:13s
epoch 135| loss: 0.24289 | train_mae: 0.27412 | valid_mae: 0.37517 |  0:00:14s
epoch 136| loss: 0.36158 | train_mae: 0.27616 | valid_mae: 0.37479 |  0:00:14s
epoch 137| loss: 0.18266 | train_mae: 0.27484 | valid_mae: 0.37374 |  0:00:14s
epoch 138| loss: 0.16764 | train_mae: 0.27203 | valid_mae: 0.37465 |  0:00:14s
epoch 139| loss: 0.15425 | train_mae: 0.27229 | valid_mae: 0.3777  |  0:00:14s
epoch 140| loss: 0.18496 | train_mae: 0.27004 | valid_mae: 0.37973 |  0:00:14s
epoch 141| loss: 0.15953 | train_mae: 0.27113 | valid_mae: 0.37869 |  0:00:14s
epoch 142| loss: 0.17338 | train_mae: 0.26777 | valid_mae: 0.3697  |  0:00:14s
epoch 143| loss: 0.18787 | train_mae: 0.26694 | valid_mae: 0.3712  |  0:00:14s
epoch 144| loss: 0.18728 | train_mae: 0.26977 | valid_mae: 0.37195 |  0:00:14s
epoch 145| loss: 0.3316  | train_mae: 0.26853 | valid_mae: 0.37204 |  0:00:14s
epoch 146| loss: 0.17949 | train_mae: 0.27007 | valid_mae: 0.37733 |  0:00:15s
epoch 147| loss: 0.20248 | train_mae: 0.26918 | valid_mae: 0.37494 |  0:00:15s
epoch 148| loss: 0.12961 | train_mae: 0.26772 | valid_mae: 0.37351 |  0:00:15s
epoch 149| loss: 0.16773 | train_mae: 0.26866 | valid_mae: 0.37851 |  0:00:15s
epoch 150| loss: 0.28433 | train_mae: 0.27112 | valid_mae: 0.38511 |  0:00:15s
epoch 151| loss: 0.29758 | train_mae: 0.27122 | valid_mae: 0.3919  |  0:00:15s
epoch 152| loss: 0.14772 | train_mae: 0.27088 | valid_mae: 0.39842 |  0:00:15s
epoch 153| loss: 0.0962  | train_mae: 0.2724  | valid_mae: 0.40553 |  0:00:15s
epoch 154| loss: 0.2514  | train_mae: 0.26993 | valid_mae: 0.40532 |  0:00:15s
epoch 155| loss: 0.22224 | train_mae: 0.26215 | valid_mae: 0.40596 |  0:00:15s
epoch 156| loss: 0.225   | train_mae: 0.27279 | valid_mae: 0.42793 |  0:00:15s
epoch 157| loss: 0.16848 | train_mae: 0.27411 | valid_mae: 0.43001 |  0:00:16s
epoch 158| loss: 0.22071 | train_mae: 0.27027 | valid_mae: 0.41793 |  0:00:16s
epoch 159| loss: 0.21576 | train_mae: 0.25671 | valid_mae: 0.40034 |  0:00:16s
epoch 160| loss: 0.21328 | train_mae: 0.25611 | valid_mae: 0.39512 |  0:00:16s
```

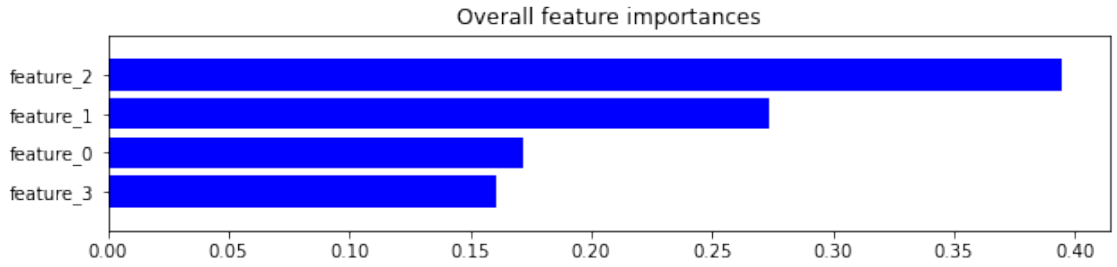
```
epoch 161| loss: 0.24389 | train_mae: 0.25582 | valid_mae: 0.39255 |  0:00:16s
epoch 162| loss: 0.17815 | train_mae: 0.24807 | valid_mae: 0.38827 |  0:00:16s
epoch 163| loss: 0.13673 | train_mae: 0.24825 | valid_mae: 0.38658 |  0:00:16s
epoch 164| loss: 0.21365 | train_mae: 0.24811 | valid_mae: 0.37979 |  0:00:16s
epoch 165| loss: 0.1076  | train_mae: 0.24731 | valid_mae: 0.37627 |  0:00:16s
epoch 166| loss: 0.22482 | train_mae: 0.2453  | valid_mae: 0.37606 |  0:00:16s
epoch 167| loss: 0.1704  | train_mae: 0.2439  | valid_mae: 0.37915 |  0:00:17s

Early stopping occurred at epoch 167 with best_epoch = 117 and best_valid_mae =
0.3674
Best weights from best epoch are automatically used!
```

[4]:
```
explainability_matrix , masks = regressor.explain(X_test)

# Normalize the importance by sample
normalized_explain_mat = np.divide(explainability_matrix, explainability_matrix.
 ↪sum(axis=1).reshape(-1, 1)+1e-8)

# Add prediction to better understand correlation between features and␣
 ↪predictions
val_preds = regressor.predict(X_test)

# explain_and_preds = np.hstack([normalized_explain_mat, val_preds.reshape(-1,␣
 ↪1)])
explain_and_preds = np.hstack([normalized_explain_mat])
```

1.4 Feature importance

As with the majority of estimators, TabNet provides access to a ranking of features in terms of their overall importance:

[5]:
```
feat_importances = regressor.feature_importances_
indices = np.argsort(feat_importances)
# plot
fig, ax = plt.subplots(figsize=(10, 2))
plt.title("Overall feature importances")
plt.barh(range(len(feat_importances)), feat_importances[indices],color="b",␣
 ↪align="center")
features = ['feature_{}'.format(i) for i in range(0, 4)]
plt.yticks(range(len(feat_importances)), [features[idx] for idx in indices])
# all features
# plt.ylim([-1, len(feat_importances)])
# Top 25 features
plt.ylim([len(feat_importances)-5, len(feat_importances)])
plt.show();
```



1.5 Local interpretability

However, the beauty of TabNet is that it allows us to not only to obtain the overall feature importances, but also inspect the importance of each of the features for each of the individual rows, here for the validation data:

[6]:
```
px.imshow(explain_and_preds[:,:],
          labels=dict(x="Features", y="Samples", color="Importance"),
          #x=features+["prediction"],
          title="Sample wise feature importance",
          color_continuous_scale='Jet',
          height=1000)
```

It is interesting to see the variation of certain feature importances along the rows. This explains more about how each feature behaves throughout the dataset, which cannot be brought out by the simple overall ranking of the features.

We can also produce a correlation matrix for the importance of the features with respect to each other

[7]:
```
explain_and_preds = np.hstack([normalized_explain_mat, val_preds.reshape(-1,␣
 ↪1)])
correlation_importance = np.corrcoef(explain_and_preds.T)
px.imshow(correlation_importance,
          labels=dict(x="Features", y="Features", color="Correlation"),
          x=features+["prediction"], y=features+["prediction"],
          title="Correlation between attention mechanism for each feature and␣
 ↪predictions",
          color_continuous_scale='Jet')
```