Среднесрочный прогноз геомагнитных бурь

Павел Чеканов
Кураторы: Марк Блуменау, Ольга Хабарова

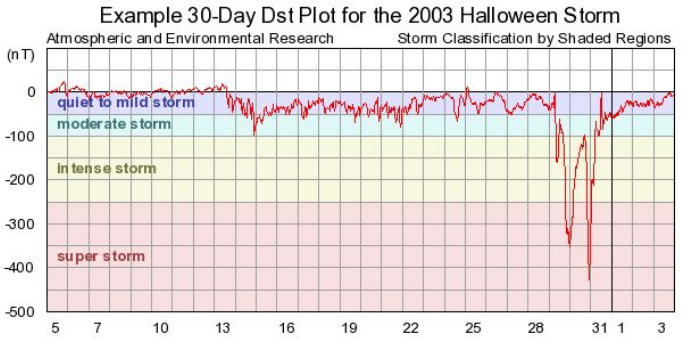
Данные

Главным источником информации являются плотность солнечного ветра получаемая со спутника и dst индекс, рассчитываемый по измерениям на земле. Индекс показывает мощность шторма - чем меньше значение тем сильнее буря

Данные собраны с https://ngdc.noaa.gov/ , dst индекс взят https://wdc.kugi.kyoto-u.ac.jp/ скриптом

- Данные из отдельных файлов собраны в датафреймы
- Убраны все потенциально лишние показатели
- dst индекс из почасового преобразован в поминутный
- Данные усреднены по минуте и слиты с dst индексом

Получилась таблица в ~200 mb



Датасет

За событие взят dst индекс меньше -30

Далее от времени начала события t взято окно [t - 40h, t-4h]

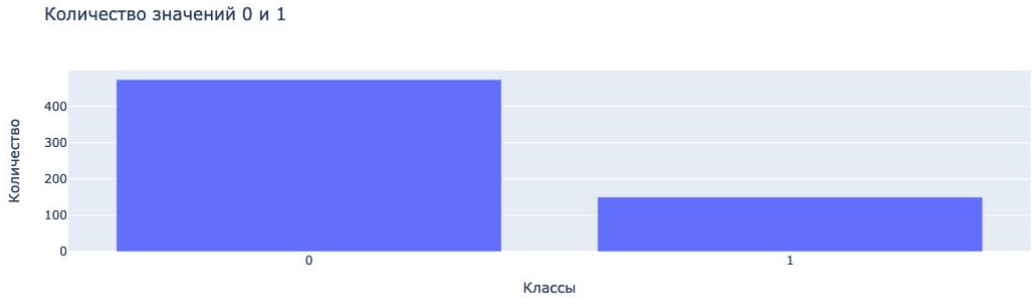
Эти данные считаем положительным классом

В оставшихся данных взяты полные 36 часовые участки

Это отрицательный класс

Недостающие записи заменены методом линейной интерполяции или bfill в случае когда первая запись NaN

Данные сбалансированы с помощью RandomOverSampling



Фурье преобразование

Применил оконное (окно Кайзера) преобразование Фурье из пакета numpy, добавил результа как фичи в датасет

$$w(n) = \frac{|I_0\left(\beta\sqrt{1-\left(\frac{2n-N+1}{N-1}\right)^2}\right)|}{|I_0(\beta)|}$$

```
for i in range(len(X)):
    sig = X.iloc[i].values
    win = np.kaiser(len(sig), 5)
    x_win = sig * win
    X_win = fftpack.fft(x_win)
    new_columns = np.vstack([sig, np.abs(X_win)]).reshape(-1)
    for j in range(len(new_columns)):
        new_X.at[i, f'new_col_{j+1}'] = new_columns[j]
```

DL часть

```python
class TimeSeriesDataset(Dataset):
    def __init__(self, features, labels):
        self.features = torch.tensor(features[:, np.newaxis, :], dtype=torch.float32)
        self.labels = torch.tensor(labels, dtype=torch.float32)

    def __len__(self):
        return len(self.features)

    def __getitem__(self, idx):
        return self.features[idx], self.labels[idx]
```

Для данных написал класс датасета, много страдал с размерностями, хотя входные данные простые (N, 12960)

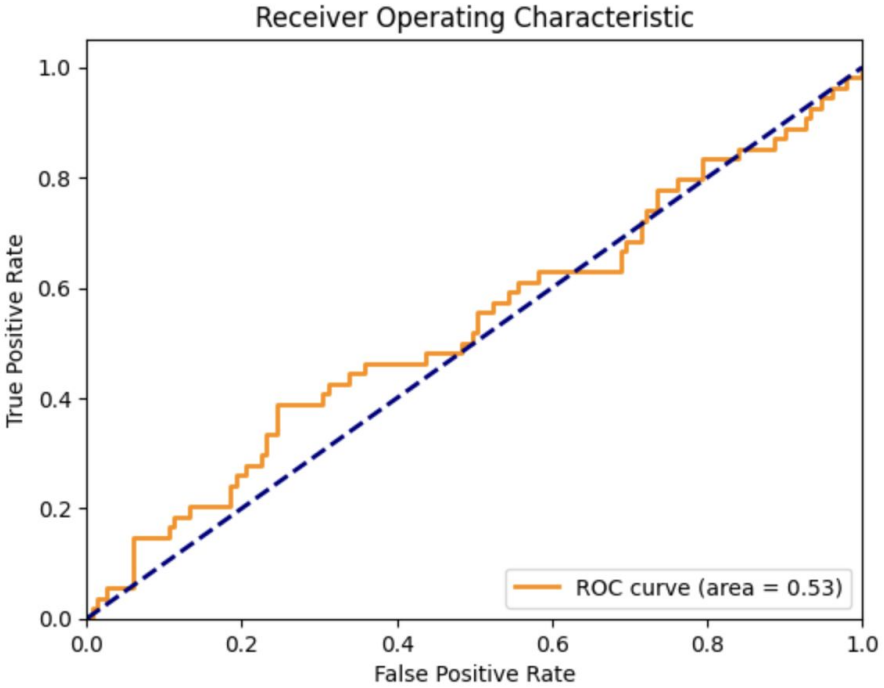
Модель

На семинаре говорили про использование 1d сверток для временных рядов

```
ComplexCNN(
  (conv1): Conv1d(1, 16, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn1): BatchNorm1d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(16, 32, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv1d(32, 64, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool3): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv4): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool4): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv5): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn5): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=103680, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=128, bias=True)
  (fc4): Linear(in_features=128, out_features=1, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

Результат

Добиться результата сравнимого с CatBoost не удалось.



CatBoost

РезультатCatBoost

```
model = CatBoostClassifier(iterations=100,
                           learning_rate=.1,
                           depth=10)

model.fit(X_train, y_train)
```

```
84:     learn: 0.0246102        total: 4m 18s   remaining: 45.6s
85:     learn: 0.0240778        total: 4m 21s   remaining: 42.6s
86:     learn: 0.0234830        total: 4m 24s   remaining: 39.5s
87:     learn: 0.0230538        total: 4m 27s   remaining: 36.5s
88:     learn: 0.0224537        total: 4m 30s   remaining: 33.4s
89:     learn: 0.0218093        total: 4m 33s   remaining: 30.4s
90:     learn: 0.0214171        total: 4m 36s   remaining: 27.4s
91:     learn: 0.0207793        total: 4m 39s   remaining: 24.3s
92:     learn: 0.0201635        total: 4m 42s   remaining: 21.3s
93:     learn: 0.0197921        total: 4m 45s   remaining: 18.2s
94:     learn: 0.0193366        total: 4m 49s   remaining: 15.2s
95:     learn: 0.0188504        total: 4m 52s   remaining: 12.2s
96:     learn: 0.0183312        total: 4m 55s   remaining: 9.13s
97:     learn: 0.0178875        total: 4m 58s   remaining: 6.09s
98:     learn: 0.0174839        total: 5m 1s    remaining: 3.05s
99:     learn: 0.0171652        total: 5m 4s    remaining: 0us

<catboost.core.CatBoostClassifier at 0x1513bd8d0>
```