Среднесрочный прогноз геомагнитных бурь

Павел Чеканов
Кураторы: Марк Блуменау, Ольга Хабарова

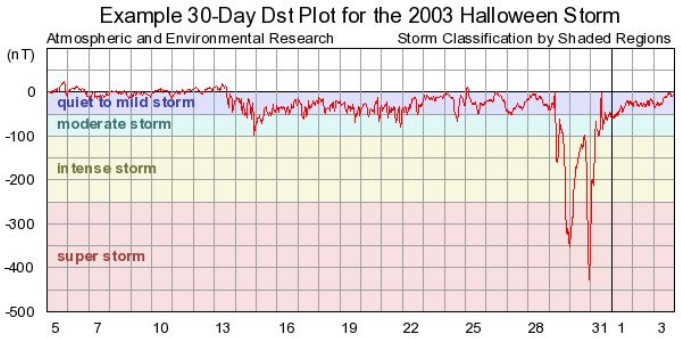
Данные

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]

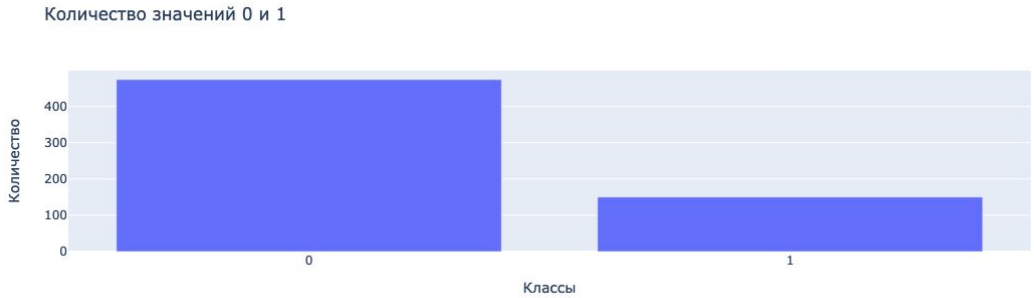
Эти данные считаем положительным классом

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



Модель

В качестве первой пробы взят CatBoostClassifier

```
preds_class = model.predict(X_test)
preds_proba = model.predict_proba(X_test)
```

```
print(accuracy_score(y_test, preds_class))
```

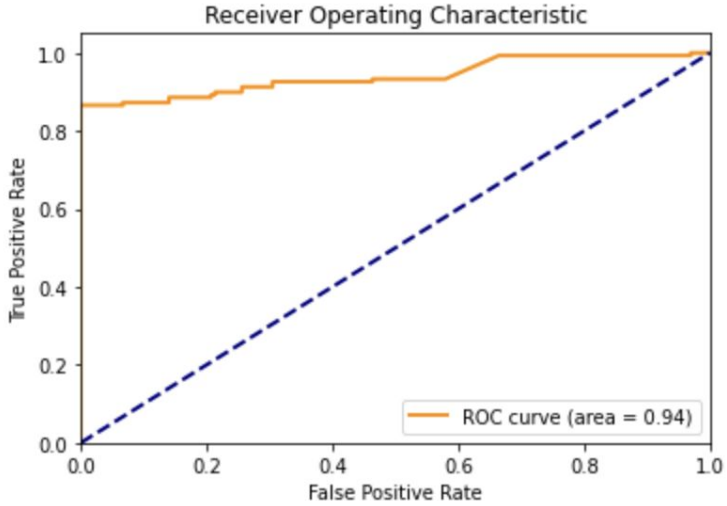
```
0.8434504792332268
```

```
print(roc_auc_score(y_test, preds_class))
```

```
0.8453920445244721
```

```
print(precision_score(y_test, preds_class))
```

```
0.8048780487804879
```



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]
```

CatBoost

Снова применил CatBoostClassifier, результат получился лучше, StandarScaler никак на результат не повлиял

```
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>
```