DOI: 10.18137/RNU.V9187.25.04.P.141 УДК 519.2:524.73:004.77

Тряпицын Валерий Леопольдович
инженер-программист, ООО «Фарватер», Санкт-Петербург, Россия. ORCID: 0009-0009-3866-2572
Электронный адрес: val20063@mail.ru
Valery L. Tryapitsyn
Software Engineer, "Seaway" LLC, Saint Petersburg, Russia. ORCID: 0009-0009-3866-2572
E-mail address: val20063@mail.ru

ПРЕДСКАЗАНИЕ ВЕРТИКАЛЬНОЙ ЭЛЕКТРОННОЙ КОНЦЕНТРАЦИИ ИОНОСФЕРЫ ДЛЯ ВЫЧИСЛЕНИЯ ИОНОСФЕРНЫХ ПОПРАВОК В РАДИОНАВИГАЦИИ ПРИ ПОМОЩИ БИБЛИОТЕК МАШИННОГО ОБУЧЕНИЯ SKLEARN

Аннотация. В данной статье рассматривается методика предсказания вертикальной электронной концентрации ионосферы для вычисления ионосферных поправок в радионавигации с использованием библиотек машинного обучения на языке Python. Проведен анализ различных алгоритмов машинного обучения, таких как Decision Tree Regressor и Random Forest, для моделирования динамики ионосферных условий. Проведено сравнение этих методов с нейронными сетями. В качестве исходных данных использовались измерения ионосферной активности и геофизические параметры. Полученные модели продемонстрировали достаточные точностные характеристики. Достигнута возможность прогнозирования в реальных условиях, что позволяет повысить точность систем спутниковой навигации и снизить влияние ионосферных искажений на измерения спутниковых сигналов. Проанализированы методы разложения изображений предсказанных распределений электронной концентрации для передачи предсказанных распределений на удаленные навигационные вычислительные устройства методом разложения на сферические гармоники и методом сплайн-интерполяции. Результаты подтверждают эффективность применения методов машинного обучения для решения задач пространственной и временной оценки ионосферных параметров в радионавигационных системах.

Ключевые слова: ионосферные поправки, электронная концентрация, предсказание, машинное обучение, Python, радионавигация, моделирование, случайный лес, Random Forest, Decision TreeRegressor, геофизические параметры, спутниковая навигация.

Для цитирования: *Тряпицын В.Л.* Предсказание вертикальной электронной концентрации ионосферы для вычисления ионосферных поправок в радионавигации при помощи библиотек машинного обучения sklearn // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ, управление. 2025. № 4. С. 141 – 159. DOI: 10.18137/RNU.V9187.25.04.P.141

PREDICTION OF VERTICAL ELECTRON DENSITY OF THE IONOSPHERE FOR CALCULATING IONOSPHERIC CORRECTIONS IN RADIO NAVIGATION USING SKLEARN MACHINE LEARNING LIBRARIES

Abstract. The article discusses a method for predicting the vertical electron density of the ionosphere for calculating ionospheric corrections in radio navigation using machine learning libraries in Python. Vari-

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

ous machine learning algorithms, such as Decision Tree Regressor and Random Forest, for modeling the dynamics of ionospheric conditions are analyzed. These methods are compared with neural networks. Measurements of ionospheric activity and geophysical parameters were used as input data. The resulting models demonstrated sufficient accuracy characteristics. The possibility of forecasting in real-world conditions was achieved, which allows for increasing the accuracy of satellite navigation systems and reducing the impact of ionospheric distortions on satellite signal measurements. Methods for decomposing images of predicted electron density distributions for transmitting the predicted distributions to remote navigation computing devices using the spherical harmonic decomposition method and spline interpolation are analyzed. The results confirm the effectiveness of machine learning methods for solving problems of spatial and temporal estimation of ionospheric parameters in radio navigation systems.

Keywords: ionospheric corrections, electron density, prediction, machine learning, Python, radio navigation, modeling, Random Forest, Decision Tree Regressor, geophysical parameters, satellite navigation.

For citation: Tryapitsyn V.L. (2025) Prediction of vertical electron density of the ionosphere for calculating ionospheric corrections in radio navigation using sklearn machine learning libraries. *Vestnik of Russian New University. Series: Complex Systems: Models, analysis, management*. No. 4. Pp. 141 – 159. DOI: 10.18137/RNU.V9187.25.04.P.141 (In Russian).

Введение

Ионосфера Земли представляет собой важный аспект для спутниковых измерений, который играет ключевую роль в радиосвязи, навигации и других технологиях, зависящих от электромагнитных волн. Одним из основных параметров, характеризующих ионосферу, является вертикальное полное электронное содержание (Vertical total electron content – VTEC), которое измеряет общее количество свободных электронов в столбе воздуха от поверхности Земли до верхней границы ионосферы. Прогнозирование VTEC[1] [1; 2] имеет важное значение для обеспечения точности измерения параметров радиосигналов и навигационных систем, таких как GPS. VTEC влияет на распространение радиоволн, их задержку и искажения, что может привести к ошибкам в навигационных системах и ухудшению качества связи.

Существует задача вычисления ионосферных поправок и применения их на мобильном устройстве, которым, как правило, является навигационный терминал. В данной работе представлена технология, которая позволяет при использовании полученного двумерного распределения VTEC в заданном географическом районе раскладывать его на коэффициенты, передавать в навигационный терминал, который, в свою очередь, будет восстанавливать распределение и вычислять ионосферные поправки к навигационным радиоизмерениям, повышая тем самым точность определения места пользователя.

Применение машинного обучения для прогнозирования VTEC

С развитием технологий и увеличением объёмов данных методы машинного обучения (далее – МО) становятся всё более популярными для прогнозирования [3; 4]. МО анализирует большие наборы данных, выявляет скрытые закономерности и составляет прогнозы на основе исторических данных. Для прогнозирования VTEC были собраны данные о солнечной активности, геомагнитных бурях, метеорологических условиях и других факторах ионосферы из таких источников, как спутники, наземные станции и модели.

[1] scikit-learn. Machine Learning in Python. URL: https://scikit-learn.org/stable/documentation.html (дата обращения: 23.10.2025).

Опишем основные этапы и задачи данной работе. Одна из задач – это передача карты или модели распределения ионосферы в виде аналитических выражений и коэффициентов на мобильное устройство для расчёта ионосферных поправок с учётом местоположения пользователя. Это включает в себя разработку компактного представления (коэффициентов разложения) карты VTEC.

Предварительная обработка данных включает очистку, нормализацию и подготовку данных путём удаления выбросов, заполнения пропусков и форматирования для анализа.

Следующий этап – выбор метода машинного обучения, например, регрессии, деревьев решений или нейронных сетей. Были протестированы деревья решений и ансамблевые случайные леса. Мы применяем ММ для прогнозирования VTEC в ионосфере Земли . Данные VTEC были получены для 59°с. ш. и 30°в. д. из Глобальной карты ионосферы (GIM) по адресу https://cddis.nasa.gov/archive/gnss/products/ionex/. Далее мы сравниваем два метода разложения: коэффициенты сферических гармоник и коэффициенты кубических сплайнов для передачи распределения VTEC в навигационные устройства.

Работа с Ionex-файлами

Для получения ионосферных данных мы используем архивные файлы Ionex. Ionex (формат обмена ионосферными данными) – стандарт обмена ионосферными данными в геодезии и навигации. Он был разработан для хранения и передачи информации об ионосферной задержке, которая может влиять на работу GPS и других систем глобального позиционирования. Файлы Ionex представляют собой текстовые файлы с заголовками и данными. Заголовки содержат информацию о времени, местоположении, типе данных и параметрах модели ионосферы. Данные содержат значения ионосферной концентрации в единицах TECU (10^{16} м$^{-2}$), представленные в виде сетки точек. Эти данные помогают корректировать сигналы GPS и навигационных систем. Файлы Ionex имеют строгую структуру, включая блоки данных со значениями задержки для различных частот и временных интервалов.

Напишем функции и утилиты для загрузки файлов IONEX, получения VTEC и построения карт VTEC. Ниже приведены примеры с использованием библиотек numpy и scipy.

Функция разбора строки файла Ionex:

```
def parse_map(tecmap, exponent=-1):
  tecmap=re.split('.*END OF TEC MAP', tecmap)[0]
  return np.stack([np.fromstring(l, sep=' ') for l in re.split('.*LAT/LON1/LON2/DLON/H\\
n',tecmap)[1:]])*10**exponent
```

Функция получения массива данных VTEC из файла Ionex:

```
def get_tecmaps(filename):
   with open(filename) as f:
      ionex=f.read()
      return [parse_map(t) for t in ionex.split('START OF TEC MAP')[1:]]
```

Функция получения значения VTEC из заданной точки координат файла Ionex:

```
MAX_LAT = 87.5 MAX_LON = 180
def get_tec(tecmap, lat, lon):
   i=round((MAX_LAT-lat)*(tecmap.shape[0]-1)/(2* MAX_LAT))
   j=round((MAX_LON  +lon)*(tecmap.shape[1]-1)/(MAX_LON*2))
```

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

return tecmap[i,j]

Функция составления строки с именем файла Ionex:

def ionex_filename(year, day, centre, zipped=True):

return'{}{:03d}0.{:02d}i{}'.format(centre, day, year%100, '.Z' if zipped else'')

Функция формирования пути к файлу:

def ionex_local_path(year, day, centre='IACG', directory=»d:/projects/PYTHON/GNSS/IONO/ionex», zipped=False):

return directory+'/'+str(year) +'/'+ionex_filename(year, day, centre, zipped)

где year и day – заданный день и год.

Функция отрисовки карты VTEC отображает карту VTEC с помощью библиотеки Matplotlib и Cartopy. Она создает график с проекцией Plate Carree, наносит цветовую карту данных TECU в диапазоне от 0 до VMAX на заданных границах долготы и широты, добавляет цветовую шкалу с подписью, форматирует оси с указанием долготы и широты, а также отображает сетку и заголовок. Эта функция позволяет визуализировать карту VTEC на земной поверхности с географическими метками и цветовой градацией TECU. VMAX – это максимальное значение данных TECU, используемое для масштабирования цветовой карты на карте VTEC. Оно определяет яркость цветов и диапазон отображаемых значений, чтобы лучше визуализировать вариации и особенности данных.

Разработка механизма получения компактного представления (в виде коэффициентов разложения) карты распределения VTEC

Построение глобальных ионосферных карт VTEC (GIM) похоже на расчет локальных ионосферных карт по измерениям на одной станции, но используется глобально распределенная сеть станций [5]. Однослойные ионосферные модели предполагают, что все свободные электроны находятся в бесконечно тонком слое на некоторой высоте над Землей. Наклонная электронная плотность (STEC) находится путем умножения VTEC на функцию отображения $mf(E)$, которая зависит от угла места E, высоты ионосферного слоя (h = 450 км) и радиуса Земли R_e. Используя метод наименьших квадратов, уточняются ионосферные задержки для всех спутников, наряду с вертикальным распределением VTEC, моделируемым как сферическое гармоническое разложение на основе широты и солнечно-фиксированной долготы. Распределение VTEC в локальной области станции зависит от географической широты φi и солнечно фиксированной долготы λi точки ионосферного прокола.

$$VTEC(\varphi,\lambda) = \sum_{n=0}^{N}\sum_{m=0}^{n}\left[A_{n,m}\Lambda(n,m)P_{n,m}(\sin\varphi)\cos(m\lambda)\right]$$

An,m – коэффициенты разложения (определяются методом наименьших квадратов);

$\Lambda(n,m)$ – нормированные функции Лежандра $\Lambda(n,m)$ – нормализующая функция;

$P_{nm}\sin(\varphi)$ – ассоциированные функции Лежандра;

Произведем вычисление задержки распространения:

$$dt = \frac{40.3}{c*f}*STEC\ ,$$

$$STEC = mf(E)*VTEC\ ,$$

$$mf(E) = \frac{1}{\cos E}*\left(1+\frac{h}{R_e}\right)^{-2}\ ,$$

где h – высота слоя ионосферы; E – угол возвышения спутника; $STEC$ – наклонная ионосферная концентрация; $mf(E)$ – функция отображения.

Напишем функцию вычисления коэффициентов сферических гармоник на языке Python:

```
def calculate_spherical_harmonic_coeffs(image_array, l_max, height, width):
# Создаем массив для хранения коэффициентов сферических гармоник,
   # размеры: (l_max+1) x (2*l_max+1), так как m варьируется от -l до l
coeffs = np.zeros((l_max + 1, 2 * l_max + 1), dtype=complex)
# Проходим по каждому пикселю изображения
for theta in range(height):
for phi in range(width):
# Значение интенсивности пикселя
r = image_array[theta, phi]
# Преобразование координат пикселя в сферические углы:
# θ (от 0 до π)
        theta_s = np.pi * (theta / height)
        # φ (от 0 до 2π)
        phi_s = 2 * np.pi * (phi / width)
# Вычисляем синус θ для учета площади элемента сферы
sin_theta = np.sin(theta_s)
# Проходим по всем l и m для вычисления соответствующих гармоник
for l in range(l_max + 1):
          for m in range(-l, l + 1):
# Вычисляем значение сферической гармоники Y_l^m в точке (θ, φ)
Y_lm = sph_harm(m, l, phi_s, theta_s)
# Накопление вклада данного пикселя в коэффициенты:
# Умножаем значение на интенсивность r и sin(θ) для учета площади
coeffs[l, m + l] += r * Y_lm * sin_theta
# Нормализация коэффициентов для приближения интеграла по всей сфере
normalization = (4 * np.pi) / (height * width)
   coeffs *= normalization
   return coeffs
```

Рассмотрим восстановление карты VTEC по коэффициентам кубического сплайна. Кубический сплайн – это гладкая кусочная функция. Она состоит из нескольких кубических полиномов, каждый из этих полиномов определён только между конкретными узлами. Сначала производится разбиение данных по узлам (точкам с заданными значениями). Для каждого интервала строится кубический полином, проходящий через соответствующие узлы. Сплайн в итоге получается таким образом, чтобы производные первая, вторая или третья, совпадали на границах интервалов, обеспечивая гладкость. Использование сплайна состоит в том, что для любой точки внутри интервала выбирается соответствующий кубический полином и вычисляется его значение:

$$S_i(x) = a_i + b_i * x + c_i * x^2 + d_i * x^3 \ ,$$

где i – номер точки.

Напишем такую функцию для расчета коэффициентов пространственного распределения VTEC через сплайн-интерполяцию:

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

```
def splineImage(image):
  # Получение размеров изображения
  height, width=image.shape
  # Создание сетки координат
  x=np.arange(width)
 y =np.arange(height)
  # Создание кубического сплайна
 spline=Rect Bivariate Spline(y, x, image)
  # Создание новой сетки для интерполяции
  x_new=np.linspace(0, width-1, width)
 y_new=np.linspace(0, height-1, height)
  image_interp=spline(y_new, x_new)
 retur nimage_interp;
```

здесь *Rect Bivariate Spline* – это класс из библиотеки SciPy, который выполняет двумерную интерполяцию по сетке. Он строит сплайн-аппроксимацию функции, заданной на прямоугольной сетке, и позволяет получать значения интерполированной функции в произвольных точках. Используя этот код, мы выбираем из папки на компьютере нужный нам файл Ionex за выбранную дату и далее мы считываем из него все значения вертикальной электронной концентрации VTEC за сутки и формируем из них массив tecmap.

```
strion=ionex_local_path(2024, 22);# 2024 – год, 22 – день года
tecmaps=get_tecmaps(strion);
tecmap=tecmaps[0];
```

Возьмем сформированный нами из файла Ionex массив значений VTEC, выведем его размерность:

```
imlen0=tecmap.shape[0];
imlen1=tecmap.shape[1];
```

Найдем коэффициенты сферических гармоник:

```
coefficients=calculate_spherical_harmonic_coeffs(tecmap, order, imlen0, imlen1)
```

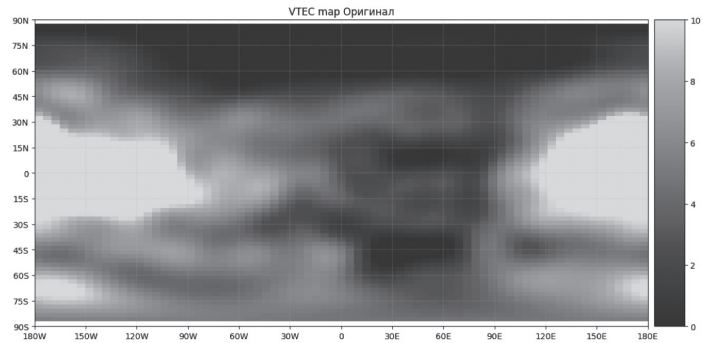
Реконструируем изображение по ним:

```
return_image_Harmonic=reconstruct_image(coefficients, imlen0, imlen1)
```

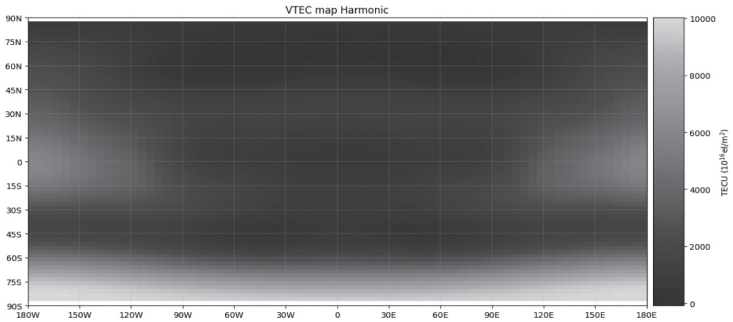
Найдем коэффициенты кубического сплайна:

```
return_image_Cubic=splineImage(tecmap);
```

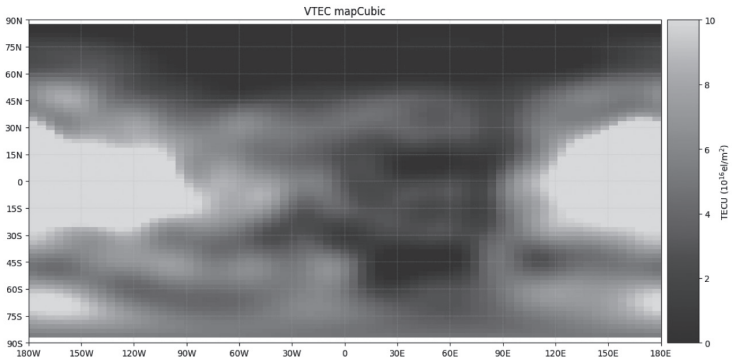
Нарисуем оригинальную карту VTEC:



Восстановим карту VTEC при помощи коэффициентов сферических гармоник:



Восстановим карту VTEC при помощи коэффициентов сплайна:



Кубическая сплайн-интерполяция точнее сферического гармонического разложения, поскольку сплайны представляют собой локальные аппроксимации, которые лучше отражают такие особенности, как края и текстуры. Сферические гармоники являются глобальными и аппроксимируют всю функцию, часто теряя локальные детали. Изображения содержат множество локальных структур, которые сплайны хорошо моделируют, в то время как гармоники подходят для гладких периодических функций. Сплайны обеспечивают регулируемую гладкость для лучшей адаптации к особенностям изображения. Гармоники зависят от количества компонентов; слишком малое их количество может привести к потере деталей. Кроме того, сплайны обеспечивают хорошую стабильность и локальные корректировки, повышая точность реконструкции изображения.

Прогнозирование электронной концентрации ионосферного слоя при помощи машинного обучения с использованием библиотеки sklearn

В качестве входных данных мы использовали измерения солнечной активности и солнечно-земных процессов, таких как поток радиоизлучения F10.7, скорость солнечного ветра, индекс Bz и индекс Dst (геомагнитные возмущения). Данные были загружены из базы данных NASA OMNI Web[2] в формате CSV, что позволило выбрать конкретные периоды

[2] Goddard Space Flight Center – Space Physics Data Facility URL:https://omniweb.gsfc.nasa.gov/form/dx1.html

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

и поля данных. Для прогнозирования электронной концентрации ионосферы были выбраны следующие поля: дата-время, день года, час дня, индекс F10.7 (солнечное радиоизлучение на длине волны 10,7 см, индикатор солнечной активности), скорость солнечного ветра (км/с), индекс Bz (нТл, измеряющий компонент магнитного поля Земли), индекс Dst (нТл, измеряющий интенсивность геомагнитной бури) и VTEC (единицы ПЭС, измеряющие полное электронное содержание в ионосфере).

Данные загружаются в Data Frame библиотеки Pandas, индексом которого служат дата и время. Необходимые пакеты включают numpy, pandas, seaborn, sklearn.tree, sklearn.model, squared_error, mean_absolute_error и metrics. Мы подготавливаем данные для прогноза VTEC на 24 часа вперёд: после сброса индекса создаём новый Data Frame со строками, начиная с 24-го числа, содержащий только столбец VTEC, назначенный VTEC_forecast

```
data_df.reset_index(inplace=True)
VTEC_forecast = data_df.loc[24::, ['VTEC(TECU)']]
```

Исследовательский анализ данных

Для оценки корреляции между признаками мы используем матрицу диаграмм рассеяния для проведения разведочного анализа данных (см. Рисунок 1). Они будут показывать визуально видимое влияние одного признака на другой по принципу «каждый с каждым». Это будет построено для каждой пары признаков по двум осям как графики. Применим для этого библиотечную функцию sns.pairplot(data_df).

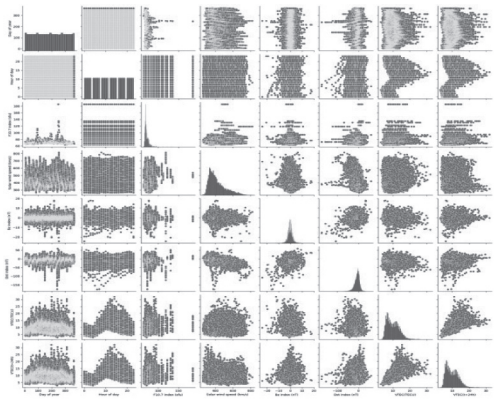


Рисунок 1. Матрица рассеяния
Источник: здесь и далее рисунки выполнены автором.

Корреляционный анализ

Произведем корреляционный анализ признаков данных. Применим для этого метод corr(). Он создает сложную матрицу, включающую в себя связи между разными признаками по принципу «каждый с каждым». На пересечении столбцов и строк, которые состоят из признаков, устанавливается число, означающее степень корреляции признаков между собой, от –1 до 1, где 0 – это отсутствие корреляции. Ячейки этой матрицы раскрашиваются в разные цвета, степень насыщенности которых – от синего до красного – пропорциональна значению корреляции (см. Рисунок 2).

В Python это можно сделать следующим образом:

c = np.round(data_df.corr(), 2)

plt.figure(figsize=(15,15))

sns.heatmap(c, annot=True, vmin=-1, vmax=1, cmap='coolwarm', square=True)

Этот код сначала вычисляет матрицу корреляции для числовых столбцов в data_df и округляет её до двух знаков после запятой.

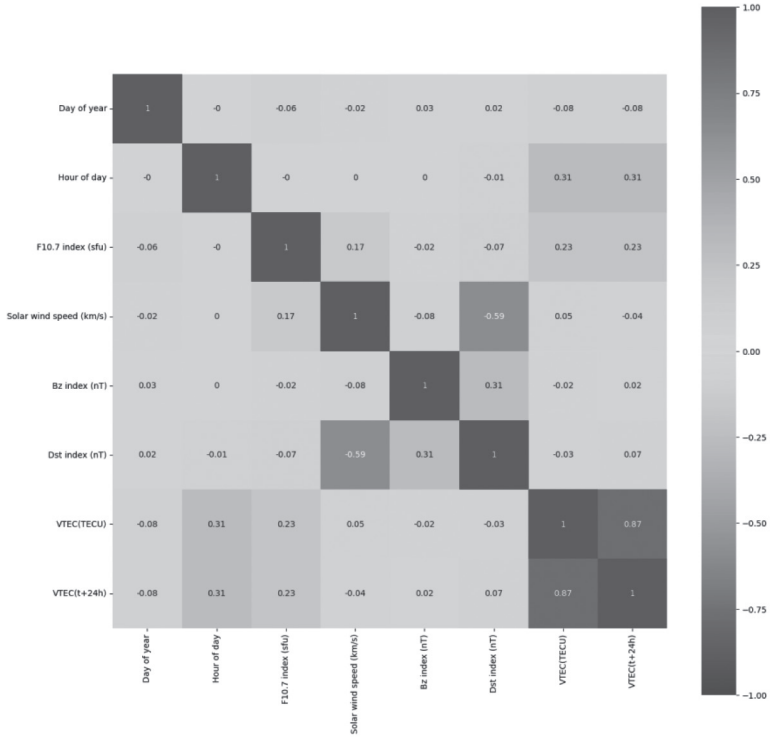


Рисунок 2. Тепловая карта корреляций параметров (Correlation Heatmap)

Найдем все корреляции с выходными данными и отсортируем

correlations=data_df.corr()['VTEC(forecast)'].sort_values()

Подготовка данных для обучения

Далее преобразуем данные в массивы NumPy, которые будут использоваться алгоритмами scikit-learn [6]. X будет матрицей, которая имеет элементы данных в последовательных строках и разные входные признаки в разных столбцах. Y будет вектором, состоящим из столбца «VTEC(forecast)».

X=data_df.drop(['VTEC(forecast)'], axis=1).to_numpy()

y=data_df ['VTEC(forecast)'].to_numpy()

dates=pd.to_datetime(data_df.index)

Мы проведем тестирование за 2023 год (с 1 января по 30 декабря), что составляет 8736 точек данных, при этом исключим тестовые данные из обучающих данных:

split_time=8736

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

```
X_train=X[:-split_time]
y_train=y[:-split_time]
time_train=dates[:-split_time]
X_test=X [-split_time:]
y_test=y [-split_time:]
time_test=dates [-split_time:]
```

Построим сводный график разделенных обучающих и тестовых данных (см. Рисунок 3):

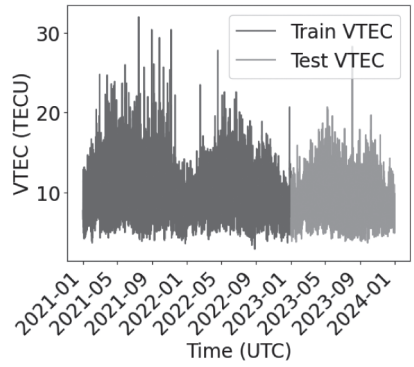


Рисунок 3. График разделенных обучающих и тестовых данных VTEC

Настройка модели Decision Tree Regressor

Мы будем использовать кросс-валидацию временных рядов по K-кратности с 20 разделениями, используя cross_val_score() и KFold для оценки различных гиперпараметров. Цель – выбрать гиперпараметры, которые максимизируют эффективность модели. Разделение данных подразумевает последовательную организацию данных временных рядов. Сначала выбираем значение K (обычно 5–10) для разделения данных на K последовательных временных окон, избегая случайных разделений и предотвращая утечку данных. Для каждого разделения модель обучается на первых N наблюдениях и тестируется на следующих M, обеспечивая чередование обучения и тестирования во времени. Этот процесс повторяется для всех K разделений, при этом каждое окно один раз служит тестовым набором. После завершения всех итераций собираются метрики производительности, такие как среднеквадратическая ошибка (RMSE) или средняя ошибка (MAE), для оценки общей эффективности модели.

Выбор модели и ее обучение

Для прогнозирования VTEC мы будем использовать Decision TreeRegressor из scikit-learn [6], реализующий алгоритм регрессии на основе дерева решений. Он прогнозирует непрерывные значения, рекурсивно разбивая данные на основе значений признаков и выбирая разбиения для минимизации ошибки прогнозирования (например, среднеквадратичной ошибки).

Параметрами модели являются:

criterion: измеряет качество разбиения («rmse» или «mae»);

max_thought: ограничивает глубину дерева для предотвращения переобучения;

min_samples_split: минимальное количество выборок для разбиения узла;

min_samples_leaf: минимальное количество выборок в листовом узле;

max_weights: количество признаков, учитываемых при разбиении.

Мы будем настраивать гиперпараметр max_depth в диапазоне значений от 1 до 20, выбирая оптимальное значение на основе наименьшего значения среднеквадратичной ошибки:

max_depth= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]

forvalinmax_depth:

score=cross_val_score(DecisionTreeRegressor(max_depth=val), X_train, y_train, cv=tscv, scoring=»neg_mean_squared_error»)

Для maxdepth = 4,rmse= 1.84

Настройки гиперпараметров Random Forest Regressor

Для max depth = 6rmse= 1.75

max_features= [1, 2, 3, 4, 5, 6, 7]

forcountinmax_features:

score=cross_val_score(RandomForestRegressor(max_features=count, max_depth=7), X_train, y_train, cv=tscv, scoring="neg_mean_squared_error")

Для max_features = 3,rmse= 1.69

estimators= [10, 50, 100, 200, 300, 400, 500, 1000]

forcountinestimators:

score=cross_val_score(RandomForestRegressor(n_estimators=count, max_features=3, max_depth=7), X_train, y_train, cv=tscv, scoring="neg_mean_squared_error")

Для estimators = 200,rmse= 1.68

Преимущества Decision Tree Regressor состоят в простоте интерпретации – деревья решений легко визуализировать и интерпретировать. Также они не требуют масштабирования данных, деревья решений не чувствительны к масштабу признаков, могут обрабатывать как числовые, так и категориальные данные.

Недостатки метода – переобучение и нестабильность. Деревья решений могут легко переобучаться, особенно если не ограничены по глубине. Небольшие изменения в данных могут привести к значительным изменениям в структуре дерева.

Напишем код обучения модели Decision Tree Regressor на Python:

model_dtree=DecisionTreeRegressor(max_depth=4)

model_dtree=model_dtree.fit(X_train, y_train)

y_pred_train=model_dtree.predict(X_train)

Результаты

Тестирование модели Decision Tree Regressor

Оценим нашу модель на тестовых данных, чтобы получить окончательную оценку точности нашей модели и выведем метрики RMSE, Explain variance score, R2 score:

Root mean squared error (RMSE): 1.42

Explain variance score = 0.77

R2 score = 0.77

Построим графики предсказанной VTEC за январь – декабрь 2023 г. Обозначим VTEC forecast – предсказанные значения (см. Рисунок 4).

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

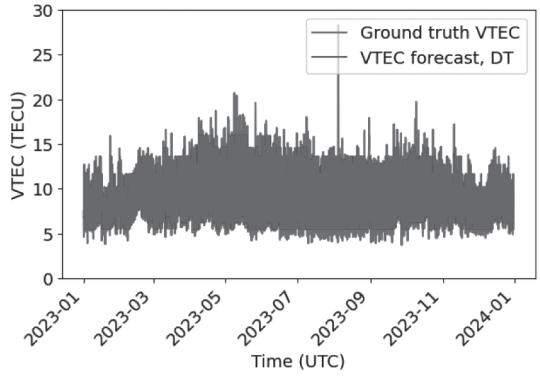


Рисунок 4. Линейный график сравнения фактическихи прогнозируемых значений VTEC

Участок на 8 дней: 23–30 декабря 2023 г. (см. Рисунок 5).

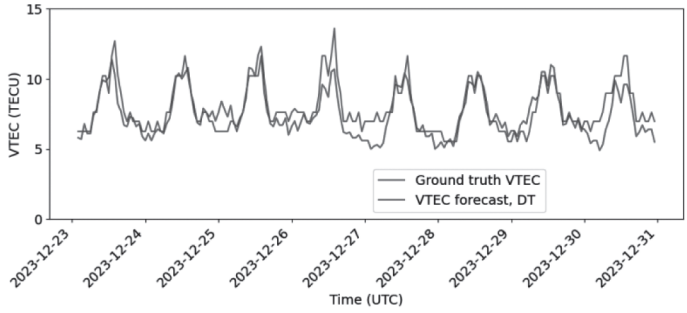


Рисунок 5. Реальные и предсказанные значения VTEC за указанный период

Использование библиотеки Random Forest

Попробуем улучшить оценку, используя ансамбль деревьев решений в форме случайного леса. Пример ниже представляет модель случайного леса с гиперпараметрами, которые минимизируют RMSE, который мы нашли ранее. Напишем код для обучения модели.

model_rfr=RandomForestRegressor(max_features=3, max_depth=7, n_estimators=500)
model_rfr=model_rfr.fit(X_train, y_train)
y_pred_train_rf=model_rfr .predict(X_train)
Root mean square error (RMSE): 1.48
Explain variance score = 0.84
R2 score = 0.84

Feature importance

Ансамбли деревьев решений, такие как случайный лес, могут использоваться для расчета оценки важности признаков. Построим гистограмму важности признаков. По ней мы видим, что основным признаком предсказания VTEC являются сами предшествующие значения VTEC, а также время дня. Построим гистограмму важности признаков (Feature Importance Bar Chart), которая отображает относительную важность каждого признака в модели. Она поможет нам отобрать важные признаки для обучения модели (см. Рисунок 6).

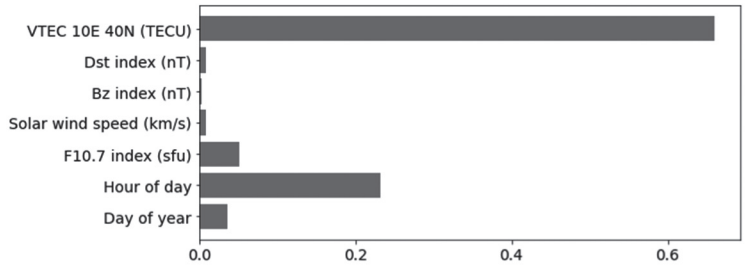


Рисунок 6. Гистограмма важности признаков (Feature Importance Bar Chart)

Тестирование модели Random Forest

Теперь оценим нашу модель на тестовых данных, чтобы получить окончательную оценку точности нашей модели и выведем метрики:

Root mean squared error (RMSE): 1.3

Explain variance score = 0.81

R2 score = 0.81

Построим те же графики, что и для Decision Tree Regressor (см. Рисунки 7, 8).

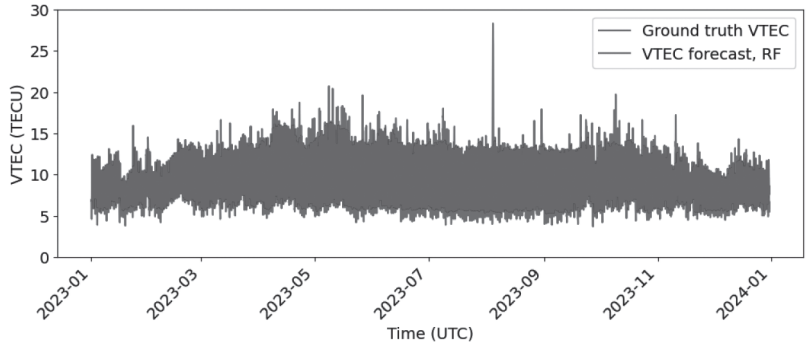


Рисунок 7. Линейный график сравнения фактических и прогнозируемых значенийVTEC

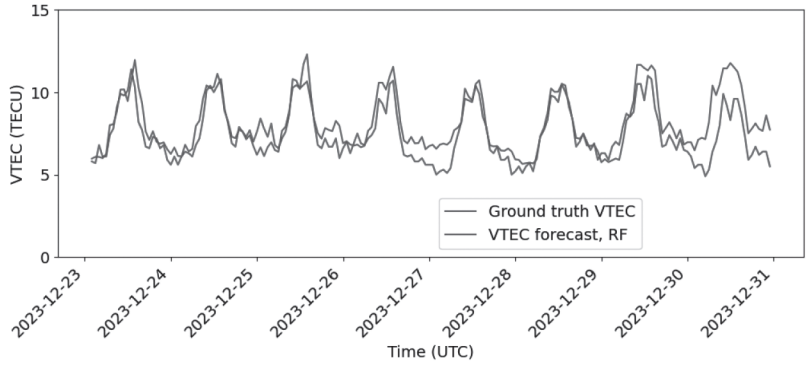


Рисунок 8. Реальные и предсказанные значения VTEC за указанный период

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

Сравнение: Decision Tree и Random Forest

Далее рассчитаем различия между реальными данными VTEC и прогнозами моделей Decision Tree и Random Forest. Используя библиотеку Seaborn, построим и сравним гистограммы двух моделей, чтобы увидеть их различия. Найдём разницу между тестовыми и прогнозируемыми данными и отобразим её на графике. Построим гистограммы распределений ошибок для моделей Decision Tree и Random Forest. Эти ошибки вычисляются как разность между прогнозируемыми и эталонными значениями. Чем уже центральный столбец, тем более точным считается прогноз. Таким образом, эти графики показывают, насколько прогнозы моделей отличаются от реальных данных (см. Рисунок 9).

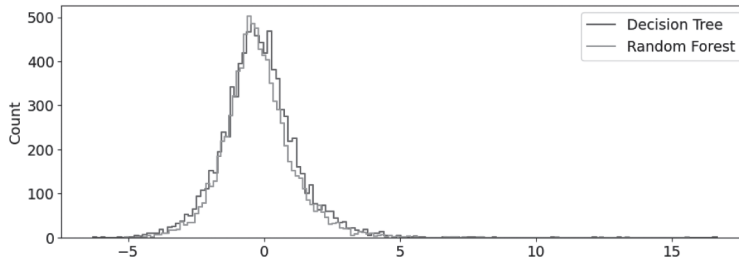


Рисунок 9. Распределение ошибок моделей через подсчет для моделей Decision Tree и Random Forest

График на Рисунке 10 обеспечивает более наглядный анализ устойчивости и смещения ошибок прогнозирования моделей. В отличие от традиционных частотных гистограмм, мы преобразуем их в кривые плотности вероятности, чтобы показать распределение ошибок между результатами прогнозирования деревьев решений и моделей случайного леса и фактическими значениями. Это преобразование достигается за счет использования плотности в качестве статистического параметра. Площадь под каждой кривой равна 1, что представляет собой распределение вероятностей ошибок, а не распределение, основанное на количестве выборок. Таким образом, мы можем более наглядно понять форму распределения ошибок различных моделей, например, определить, какая модель имеет более концентрированные или более рассеянные ошибки и в каких областях ошибки сосредоточены в основном. Таким образом, второй график стал важным инструментом для анализа смещения и стабильности прогнозирования моделей (см. Рисунок 10).

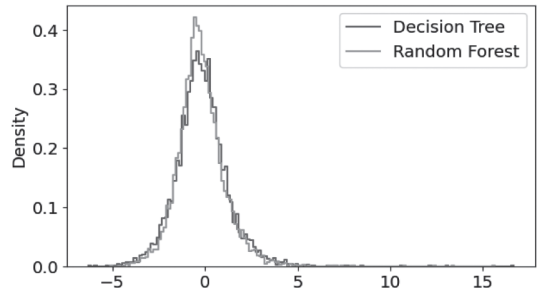


Рисунок 10. Распределение ошибок моделей через плотность распределения для моделей DecisionTree и RandomForest

Мы построили столбчатую диаграмму, сравнивающую среднеквадратичную ошибку (далее – СКО) двух моделей машинного обучения, дерева решений и случайного леса, на обучающем и тестовом наборах. Результаты показывают, что СКО дерева решений на обучающем наборе составляет приблизительно 0,15 TECU, тогда как на тестовом – приблизительно 0,20 TECU, что указывает на ограниченную предсказательную способность. Напротив, случайный лес показывает ещё лучшие результаты: СКО составляет приблизительно 0,10 TECU на обучающем наборе и снижается до 0,12 TECU на тестовом наборе, что свидетельствует о более высокой обобщающей способности. Поскольку более низкое СКО указывает на более точные прогнозы модели, можно сделать вывод, что случайный лес превосходит дерево решений по предсказательной эффективности. Ось Y на рисунке обозначена как «СКО (TECU)», что наглядно демонстрирует преимущества случайного леса с точки зрения точности и стабильности (см. Рисунок 11).

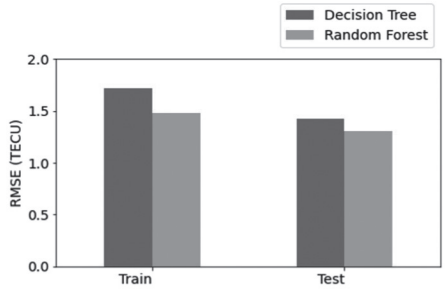


Рисунок 11. Сравнение метрик машинного обучения для моделей Decision Tree и Random Forest

Сравнивая распределения ошибок, мы наблюдали схожие тенденции в эффективности моделей дерева решений и случайного леса, с некоторым перекрытием диапазонов распределения ошибок. Гистограммы показывают, что ошибки модели случайного леса более концентрированы и уже со средней ошибкой приблизительно 2 TECU и стандартным отклонением приблизительно 5 TECU, что указывает на большую стабильность и точность её прогнозов. Модель дерева решений имеет более широкий диапазон ошибок со средним значением приблизительно 3 TECU и стандартным отклонением приблизительно 7 TECU, что указывает на меньшую стабильность и немного меньшую точность её прогнозов. Мы решили взять для практического применения модель Random Forest потому, что у нее распределение ошибок более близко к нулю, хотя средняя ошибка прогноза у обеих моделей 2-3 TECU и большинство прогнозов имело ошибку +/-1 TECU. Таким образом, модель Random Forest дает большую стабильность прогноза.

Обоснование выбора моделей и сравнение вычислительной сложности и эффективности модели Random Forest и моделей нейронных сетей

Произведем сравнение объема вычислений моделей случайного леса и модели на основе нейронных сетей, в качестве которой возьмем ConvLSTM (Convolutional Long Short-Term Memory) [7; 8]. Это разновидность рекуррентных нейронных сетей, которая сочетает в себе свойства сверточных слоев и LSTM и широко используется для обработки последовательных изображений или геопространственных временных данных.

Размер глобальной карты в градусах – это 180 по широте (от –90 до 90) и 360 по долготе (от –180 до 180). Для анализа выберем локальный участок, ограниченный заданны-

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

ми диапазонами широты и долготы, который определяет площадь региона. Определим размеры выбранных регионов и внесем в таблицу. Также вычислим количество точек для каждого района и определим вычисленное для них количество операций. Первая таблица – для модели Random Forest. В ней предсказание делается отдельно по каждой точке выбранного географического района (см. Таблицу 1).

Таблица 1

Расчет объема вычислений модели RandomForest

Район	ΔLat	ΔLon	Количество точек (примерно)	Операций (примерно)
1	30	60	(30/180·180) · (60/360 · 360) = 1800 точек	1800 ·440 = 792000
2	5	20	5 · 20 = 100 точек	44000
3	5	5	5 · 5 = 25 точек	11000

Источник: здесь и далее таблицы составлены автором.

Примем общее количество операций как количество точек × операций на точку. Произведем расчет числа операций на предсказание одной точки – это около 440 операций. Тогда для каждого района составим таблицу.

Общий объем по всем районам:

792000 + 44000 + 11000 = 847000 операций.

Сделаем то же самое для модели ConvLSTM (см. Таблицу 2). Обработка всей области за один проход через сверточную сеть – это примерно 1.7×10^6 операций для всей карты 1000×1000. 64800 – это общее количество точек (пикселей) на всей карте, используемой в моделировании.

Таблица 2

Расчет объема вычислений модели Conv LSTM

Район	ΔLat	ΔLon	Площадь, доли всей карты	Оценка по количеству точек	Операции (примерно)
1	30	60	(30/180) · (60/360) = 0,083	0,083 · 64800 ≈ 5380 точек	5380/64800 × 1,7 · $10^9 \approx 1{,}4 \cdot 10^8$
2	5	20	≈ (5/180) × (20/360) = 0,0077	0,0077· 64800 ≈ 500 точек	≈1,3· 108
3	5	5	~ 0,00077	≈ 50 точек	≈1,3 · 107

Сумма всех операций:

$1{,}4 \cdot 10^8 + 1{,}3 \cdot 110^8 + 1{,}3 \cdot 10^7 \approx 3{,}0 \cdot 10^8$.

Общий объем вычислений – порядка 10^8 операций, что существенно больше, чем у первой модели.

Произведем итоговое сравнение. Для модели Random Forest производится предсказание по каждой точке региона – это около 847000 операций для всех выбранных районов, поскольку предсказание осуществляется отдельно для каждой точки. Для модели ConvLSTM производится обработка всей карты за один проход – примерно 30 миллионов операций для всех районов, это в разы больше, несмотря на чуть большее начальное количество операций при полном анализе всей карты. Несмотря на большее количество

операций у ConvLSTM при обработке всего региона, она может быть быстрее и эффективнее за счет параллельных вычислений и обработки всей карты сразу. Модель на основе деревьев требует больше времени на предсказания при больших данных из-за необходимости обработки каждой точки отдельно. Далее произведем сравнение качество предсказаний при помощи метрик (см. Таблицу 3).

Таблица 3

Сравнение метрик машинного обучения моделей RandomForest и ConvLSTM

Метрика	Random Forest	ConvLSTM	Разница	Выводы
MAE	12,5	9,08	3,4	Модель ConvLSTM показывает на 27 % лучше по MAE
RMSE	300,2	264,0	35,8	Улучшение RMSE более чем на 11 %

Оценим чувствительность и стабильность моделей. Модель на основе деревьев стабильна при малых объемах данных, хорошо интерпретируема, но менее точна и менее гибкая к изменяющимся условиям. ConvLSTM чувствительна к параметрам обучения (число слоев, размер ядра, скорость обучения), требует тщательной настройки. Однако при правильной настройке показывает устойчивые результаты и высокую точность. Оценим работу с параметрами моделей. Для деревьев важны глубина, число деревьев, критерий разбиения, для ConvLSTM – число слоев, размер ядра, функции активации, параметры регуляризации. Их настройка должна проводиться с учетом валидационных данных и чувствительности к изменению условий.

Подведем итог сравнения наших моделей обучения. Для стабильных и быстрых решений предпочтительны модели на основе деревьев, однако для более точного анализа больших территорий лучше использовать ConvLSTM с учетом требований к вычислительным ресурсам и регулярному обновлению. Внедрение в реальные системы требует системной автоматизации, регулярного обновления данных и мониторинга качества.

Выводы

В данной работе мы произвели исследование методик прогнозирования вертикальной электронной концентрации (VTEC) в ионосфере. Основная цель этой методики – получить точные значения VTEC для расчета ионосферных поправок. Таким образом, при помощи более точного определения VTEC мы получаем более точные ионосферные поправки и более точное местоопределение в радионавигации. Этапы исследования заключались в разработке сбора данных, влияющих на прогнозирование VTEC, их обработку, включая нормализацию и исключение выбросов. Мы получили более качественные и согласованные данные. Для сравнения мы выбрали две модели машинного обучения – Decision Tree и Random Forest. Для оценки эффективности модели мы использовали среднюю абсолютную ошибку (MAE) и среднеквадратичную ошибку (RMSE).

В результате исследований мы выяснили, что модель Random Forest более точно и стабильно работает для нашего случая прогнозирования VTEC, когда динамичная по своей природе ионосферная среда часто меняется по своему составу во времени и пространстве из-за воздействия различных возмущений, таких как солнечная активность, геомагнитные бури и другие факторы. Это создает сложности для стабильной работы модели, так как она обучается на данных, которые могут устаревать. Поэтому вариативность параметров

Вестник Российского нового университета
Серия «Сложные системы: модели, анализ и управление». 2025. № 4

требует регулярного обновления модели. И, конечно, важным аспектом является сбор актуальной обучающей выборки, которая отражает текущие условия. Далее мы провели сравнение двух типов моделей МО, основанных на деревьях решений и нейронных сетях. Из этого сравнительного анализа сделали основной вывод, что для стабильных и быстрых решений предпочтительны модели на основе деревьев, однако для более точного анализа больших территорий лучше использовать нейронные сети.

Для реального использования модели в радионавигационных системах также необходимо выполнить некоторые другие шаги. С точки зрения интеграции моделей в инфраструктуру нужно создавать сервисы для автоматической передачи данных о текущей ионосфере и получения предсказаний. При этом нужно обеспечить постоянный и своевременный сбор актуальных данных (например, от спутников, наземных станций), их качество, а также внедрение системы мониторинга качества предсказаний для своевременного выявления ухудшений и использование методов онлайн-обучения или дообучения, позволяющих адаптировать модель без полного переобучения.

Литература

1. *Mannucci A.J., Wilson B.D., Yuan D.N., Ho C.H., Lindqwister U.J., Runge T.F.* A global mapping technique for GPS-derived ionospheric total electron content measurements // Radio Science. 1998. Vol. 33. No. 3. Pp. 565–582. DOI: 10.1029/97RS02707

2. *Klobuchar J.A.* Ionospheric Time-Delay Algorithm for Single-Frequency GPS Users // IEEE Transactions on Aerospace and Electronic Systems. 1987. Vol. AES-23. No. 3. Pp. 325–331. DOI: 10.1109/TAES.1987.310829

3. *Кугаевских А.В., Муромцев Д.И., Кирсанова О.В.* Классические методы машинного обучения. СПб. : Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, 2022. URL: https://books.ifmo.ru/file/pdf/3075.pdf (accessed 27.09.2025). EDN MJQVRZ.

4. *Мохамед Али Рефаее Абделлах.* Прогнозирование характеристик трафика для сетей 5 G на основе технологий искусственного интеллекта : Дис. … канд. техн. наук : 2.2.15. Санкт-Петербург, 2022. 146 с.

5. *Tryapitsyn V.L., Dubinko T.Yu.* Development and implementation of a method for transferring the parameters of ionospheric models for building a map of the electron concentration of the ionosphere in real time // AIP Conference Proceedings. 2021. Vol. 2402. No. 1. Article no. 020055. DOI: 10.1063/5.0071538. EDN HSMHJD.

6. *Бызов А.Н., Петров Ю.В., Рогожин В.А.* Применение нейронных сетей для определения дальности до источника радиоизлучения // Вопросы радиоэлектроники. 2019. № 6. С. 13–17. DOI: 10.21778/2218-5453-2019-6-13-17. EDN OGJAXO.

7. *Лимановская О.В., Алферьева Т.И.* Основы машинного обучения : Учебное пособие. Екатеринбург : Изд-во Уральского ун-та, 2020. 88 с. ISBN 978-5-7996-3015-7.

8. *Флах П.* Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / Пер. с англ. А.А. Слинкина. 2-е изд. М. : ДМК Пресс, 2015. 399 с. ISBN 978-5-97060-273-7.

References

1. Mannucci A.J., Wilson B.D., Yuan D.N., Ho C.H., Lindqwister U.J., Runge T.F. (1998) A global mapping technique for GPS-derived ionospheric total electron content measurements. *Radio Science.* Vol. 33. No. 3. Pp. 565–582. DOI: 10.1029/97RS02707

2. Klobuchar J.A. (1987) Ionospheric Time-Delay Algorithm for Single-Frequency GPS Users. In: *IEEE Transactions on Aerospace and Electronic Systems*. Vol. AES-23. No. 3. Pp. 325–331. DOI: 10.1109/TAES.1987.310829

3. Kugaevskikh A.V., Muromtsev D.I., Kirsanova O.V. (2022) *Klassicheskie metody mashinnogo obucheniya* [Classical Machine Learning Methods] : Study Guide. St. Petersburg : ITMO University Publ. 53 p. URL: https://books.ifmo.ru/file/pdf/3075.pdf (accessed 27.09.2025). (In Russian).

4. Mohamed Ali Refaee Abdellah (2022) *Predicting Traffic Characteristics for 5G Networks Based on Artificial Intelligence Technologies* : Ph.D. Diss. (Technical Sciences) : 2.2.15. St. Petersburg. 146 p. (In Russian).

5. Tryapitsyn V.L., Dubinko T.Yu. (2021) Development and implementation of a method for transferring the parameters of ionospheric models for building a map of the electron concentration of the ionosphere in real time. In: *AIP Conference Proceedings*. Vol. 2402. No. 1. Article no. 020055. DOI: 10.1063/5.0071538 (In Russian).

6. Byzov A.N., Petrov Yu.V. Rogozhin V.A. (2019) Application of Neural Networks for Determining the Range to a Radio Source. *Issues of Radio Electronics*. No 6. Pp. 13–17. DOI: 10.21778/2218-5453-2019-6-13-17 (In Russian).

7. Limanovskaya O.V., Alferieva T.I. (2020) *Osnovy mashinnogo obucheniya* [Machine Learning Fundamentals] : Study Guide. Ekaterinburg : Ural University Publ. 88 p. ISBN 978-5-7996-3015-7. (In Russian).

8. Flach P. (2012) *Machine learning. The Art and Science of Algorithms that Make Sense of Data*. Cambridge, UK ; New York : Cambridge University press. 396 p. ISBN 1107096391.

Поступила в редакцию: 31.10.2025	Received: 31.10.2025
Поступила после рецензирования: 21.11.2025	Revised: 21.11.2025
Принята к публикации: 04.12.2025	Accepted: 04.12.2025