|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Проектирование аналитических систем»

**Практическое занятие № 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Тетерин Николай Николаевич, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА 3 3](#_Toc212194522)

[Введение 3](#_Toc212194523)

[Вывод: 14](#_Toc212194524)

[Список использованных источников и литературы: 14](#_Toc212194525)

ПРАКТИЧЕСКАЯ РАБОТА 3

Введение

**Цель**

Разработать внутреннюю логику работы программы

**Модульная структура программы**

Листинг 1 – Модуль обработки данных

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.preprocessing import StandardScaler  from sklearn.impute import SimpleImputer  def load\_and\_preprocess\_data(file\_path):  data = pd.read\_csv(file\_path)  # Обработка временных меток  data['time'] = pd.to\_datetime(data['time'], errors='coerce')  data = data.dropna(subset=['is\_storm'])  data['is\_storm'] = data['is\_storm'].astype(int)  # Удаление полностью пустых колонок  data = data.dropna(axis=1, how='all')  # Нормализация числовых признаков  numeric\_cols = data.select\_dtypes(include=[np.number]).columns.tolist()  if 'is\_storm' in numeric\_cols:  numeric\_cols.remove('is\_storm')  if len(numeric\_cols) > 0:  imputer = SimpleImputer(strategy='median')  data[numeric\_cols] = imputer.fit\_transform(data[numeric\_cols])  scaler = StandardScaler()  scaled\_features = scaler.fit\_transform(data[numeric\_cols])  df\_processed = pd.DataFrame(scaled\_features, columns=numeric\_cols, index=data.index)  else:  df\_processed = pd.DataFrame(index=data.index)  # Добавляем время и целевую переменную  df\_processed['time'] = data['time']  df\_processed['is\_storm'] = data['is\_storm'].astype(int)  return df\_processed |

Листинг 2 – Модуль анализа данных

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.model\_selection import TimeSeriesSplit  from sklearn.linear\_model import LogisticRegression  from sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_auc\_score  import warnings  warnings.filterwarnings("ignore")  def create\_features(df, forecast\_horizon=1):  df = df.copy()  df = df.sort\_values('time')    # Временные признаки  df['hour'] = df['time'].dt.hour  df['day\_of\_week'] = df['time'].dt.dayofweek    # Лаговые признаки  for lag in [1, 2, 3]:  for col in ['wind\_speed\_10m', 'wind\_gusts\_10m', 'temperature\_2m']:  if col in df.columns:  df[f'{col}\_lag\_{lag}'] = df[col].shift(lag)    # Скользящие средние  for col in ['wind\_speed\_10m', 'wind\_gusts\_10m', 'precipitation']:  if col in df.columns:  df[f'{col}\_mean\_3h'] = df[col].rolling(3, min\_periods=1).mean()    # Целевая переменная с прогнозом на N часов вперёд  df['target'] = df['is\_storm'].shift(-forecast\_horizon)  df = df.dropna()    return df  def train\_and\_evaluate\_models(df\_processed, forecast\_horizons=[2, 4, 6]):  results = []    for horizon in forecast\_horizons:  df\_features = create\_features(df\_processed, forecast\_horizon=horizon)    if df\_features.shape[0] < 5:  continue    X = df\_features.drop(columns=['time', 'is\_storm', 'target'], errors='ignore')  y = df\_features['target'].astype(int)    # Кросс-валидация для временных рядов  n\_splits = min(5, max(2, df\_features.shape[0] // 10))  n\_splits = min(n\_splits, df\_features.shape[0] - 1)  if n\_splits < 2:  n\_splits = 2    tscv = TimeSeriesSplit(n\_splits=n\_splits)  model = LogisticRegression(max\_iter=1000, class\_weight='balanced', solver='lbfgs')    f1\_scores, precision\_scores, recall\_scores, auc\_scores = [], [], [], []  all\_y\_test, all\_y\_pred\_proba = [], []    for train\_idx, test\_idx in tscv.split(X):  X\_train, X\_test = X.iloc[train\_idx], X.iloc[test\_idx]  y\_train, y\_test = y.iloc[train\_idx], y.iloc[test\_idx]    if len(np.unique(y\_train)) < 2:  continue    model.fit(X\_train, y\_train)  y\_pred\_proba = model.predict\_proba(X\_test)[:, 1]    all\_y\_test.extend(y\_test)  all\_y\_pred\_proba.extend(y\_pred\_proba)    try:  y\_pred = (y\_pred\_proba > 0.5).astype(int)  f1\_scores.append(f1\_score(y\_test, y\_pred))  precision\_scores.append(precision\_score(y\_test, y\_pred))  recall\_scores.append(recall\_score(y\_test, y\_pred))  auc\_scores.append(roc\_auc\_score(y\_test, y\_pred\_proba))  except:  continue    if len(f1\_scores) > 0:  results.append({  "horizon": horizon,  "f1": np.mean(f1\_scores),  "precision": np.mean(precision\_scores),  "recall": np.mean(recall\_scores),  "auc": np.mean(auc\_scores),  "model": model,  "X": X,  "y": y,  "all\_y\_test": all\_y\_test,  "all\_y\_pred\_proba": all\_y\_pred\_proba  })    return results |

Листинг 3 – Модуль подготовки данных для Power BI

|  |
| --- |
| import pandas as pd  def create\_powerbi\_dataset(results, df\_processed):  powerbi\_data = []  for res in results:  horizon = res['horizon']  model = res['model']  X = res['X']  if hasattr(model, "predict\_proba") and len(X) > 0:  probs = model.predict\_proba(X)[:, 1]  preds = (probs > 0.5).astype(int)  # Создаем временные метки для прогнозов  start\_idx = len(df\_processed) - len(probs)  if start\_idx < 0:  start\_idx = 0  forecast\_times = df\_processed['time'].iloc[start\_idx:start\_idx + len(probs)].reset\_index(drop=True)  for i, (time, prob, pred) in enumerate(zip(forecast\_times, probs, preds)):  actual\_storm = res['y'].iloc[i] if i < len(res['y']) else None  powerbi\_data.append({  'datetime': time,  'forecast\_horizon': f'{horizon}\_hours',  'storm\_probability': round(float(prob \* 100), 2),  'storm\_prediction': int(pred),  'storm\_warning': 'ДА' if pred == 1 else 'НЕТ',  'actual\_storm': int(actual\_storm) if actual\_storm is not None else None  })  return pd.DataFrame(powerbi\_data)  def prepare\_data\_for\_powerbi(df):  df\_clean = df.copy()  if 'datetime' in df\_clean.columns:  df\_clean['datetime'] = pd.to\_datetime(df\_clean['datetime'])  numeric\_columns = ['storm\_probability', 'storm\_prediction', 'actual\_storm']  for col in numeric\_columns:  if col in df\_clean.columns:  if df\_clean[col].dtype == 'object':  df\_clean[col] = df\_clean[col].astype(str).str.replace(',', '.')  df\_clean[col] = pd.to\_numeric(df\_clean[col], errors='coerce')  return df\_clean  def export\_weather\_data(df\_processed):  """Экспорт исходных метеоданных для Power BI"""  latest\_data = df\_processed[['time', 'wind\_speed\_10m', 'wind\_gusts\_10m',  'temperature\_2m', 'precipitation', 'is\_storm']].copy()  latest\_data.rename(columns={'time': 'datetime'}, inplace=True)    for col in ['wind\_speed\_10m', 'wind\_gusts\_10m', 'temperature\_2m', 'precipitation']:  if col in latest\_data.columns:  latest\_data[col] = pd.to\_numeric(latest\_data[col], errors='coerce')    if 'is\_storm' in latest\_data.columns:  latest\_data['is\_storm'] = latest\_data['is\_storm'].astype(int)    return latest\_data |

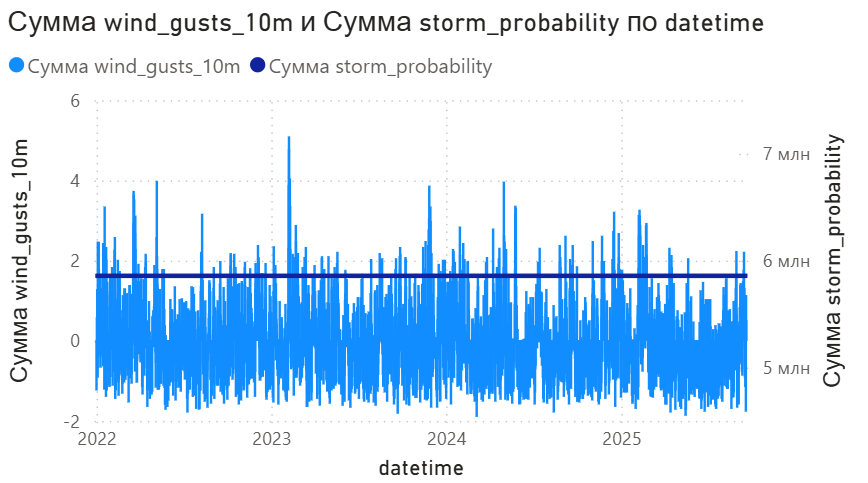
Листинг 4 – Основной файл

|  |
| --- |
| import pandas as pd  import numpy as np  from data\_processing import load\_and\_preprocess\_data  from data\_analysis import train\_and\_evaluate\_models, create\_features  from powerbi\_export import create\_powerbi\_dataset, prepare\_data\_for\_powerbi, export\_weather\_data  def main():  # 1. Загрузка и обработка данных  print("1. Загрузка и обработка данных...")  df\_processed = load\_and\_preprocess\_data("storm\_data.csv")  print(f"Данные загружены: {df\_processed.shape[0]} строк, {df\_processed.shape[1]} колонок")    # 2. Анализ данных и обучение моделей  print("2. Обучение моделей для разных горизонтов прогноза...")  forecast\_horizons = [2, 4, 6]  results = train\_and\_evaluate\_models(df\_processed, forecast\_horizons)  print(f"Обучено моделей: {len(results)}")    # 3. Вывод результатов  print("\n3. Результаты обучения:")  for res in results:  print(f"Горизонт {res['horizon']}ч: F1={res['f1']:.3f}, AUC={res['auc']:.3f}")    # 4. Подготовка данных для Power BI  print("4. Подготовка данных для визуализации в Power BI...")    # Создание датасета с прогнозами  powerbi\_df = create\_powerbi\_dataset(results, df\_processed)  powerbi\_df\_clean = prepare\_data\_for\_powerbi(powerbi\_df)    # Создание датасета с исходными данными  weather\_data = export\_weather\_data(df\_processed)    # 5. Экспорт данных  powerbi\_df\_clean.to\_csv(  'storm\_forecasts\_powerbi.csv',  index=False,  encoding='utf-8-sig',  float\_format='%.2f',  sep=';'  )    weather\_data.to\_csv(  'weather\_data\_powerbi.csv',  index=False,  encoding='utf-8-sig',  float\_format='%.2f',  sep=','  )    print("Файлы для Power BI успешно созданы:")  print("- storm\_forecasts\_powerbi.csv (прогнозы)")  print("- weather\_data\_powerbi.csv (исходные данные)")  print(f"Количество записей в прогнозах: {len(powerbi\_df\_clean)}")    # 6. Демонстрация прогнозов  print("\n5. Текущие прогнозы:")  for res in results:  model = res['model']  X\_res = res['X']  if X\_res.shape[0] > 0 and hasattr(model, "predict\_proba"):  latest\_point = X\_res.iloc[[-1]]  prob = model.predict\_proba(latest\_point)[0][1]  alert = prob > 0.5  print(f"Прогноз на {res['horizon']} часа вперёд: {prob:.2%} {'ДА' if alert else 'НЕТ'}")    print("\n=== ПРОГРАММА ЗАВЕРШЕНА ===")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

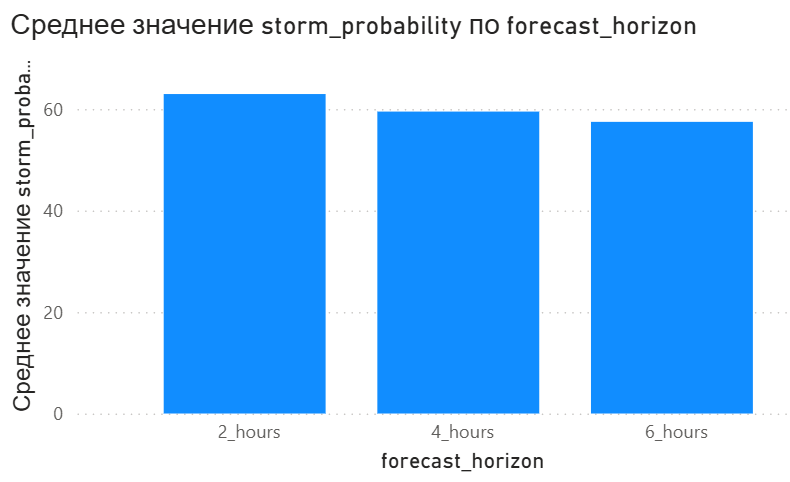
**Взаимодействие модулей**

1. main.py импортирует функции из всех модулей
2. Последовательность выполнения:

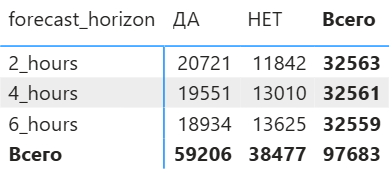
* Загрузка и предобработка данных (data\_processing)
* Создание признаков и обучение моделей (data\_analysis)
* Подготовка данных для визуализации (powerbi\_export)
* Экспорт результатов в CSV файлы



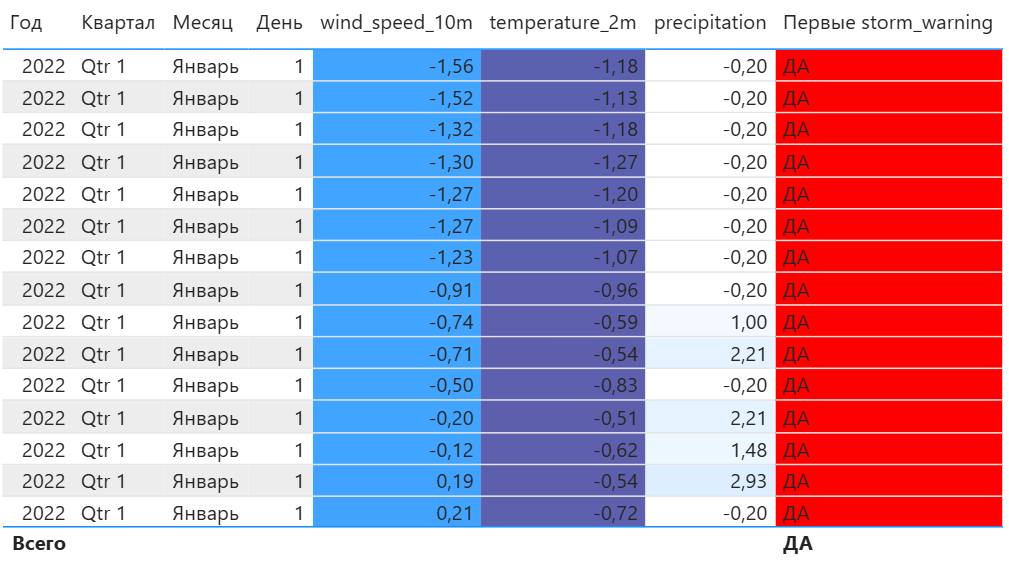
**Рисунок 1 – График временного ряда скорости ветра**



**Рисунок 2 – Столбчатая диаграмма**



**Рисунок 3– Матрица точности прогнозов**



**Рисунок 3 – Тепловая карта**

Вывод:

В ходе практической работы была разработана архитектура аналитической системы штормового предупреждения.

Список использованных источников и литературы:

1. Ростовцев В.С. Искусственные нейронные сети,   
   Издательство "Лань", 2019. — 216 с. — URL: https://e.lanbook.com/book/122180
2. Араки М. Манга: Машинное обучение,   
   Издательство "ДМК Пресс", 2020. — 214 с. — URL: <https://e.lanbook.com/book/179473>
3. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2022. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/508804