

# Выпускная квалификационная работа по курсу «Data Science PRO»

Реализация методов и алгоритмов машинного обучения для  
детектирования аварийной ситуации беспилотного летательного  
аппарата мультироторного типа по показаниям инерциальной  
навигационной системы в условиях отложенных данных.

Тарасов Матвей Андреевич



ЦЕНТР  
ДОПОЛНИТЕЛЬНОГО  
ОБРАЗОВАНИЯ  
МГТУ им. Н.Э. Баумана



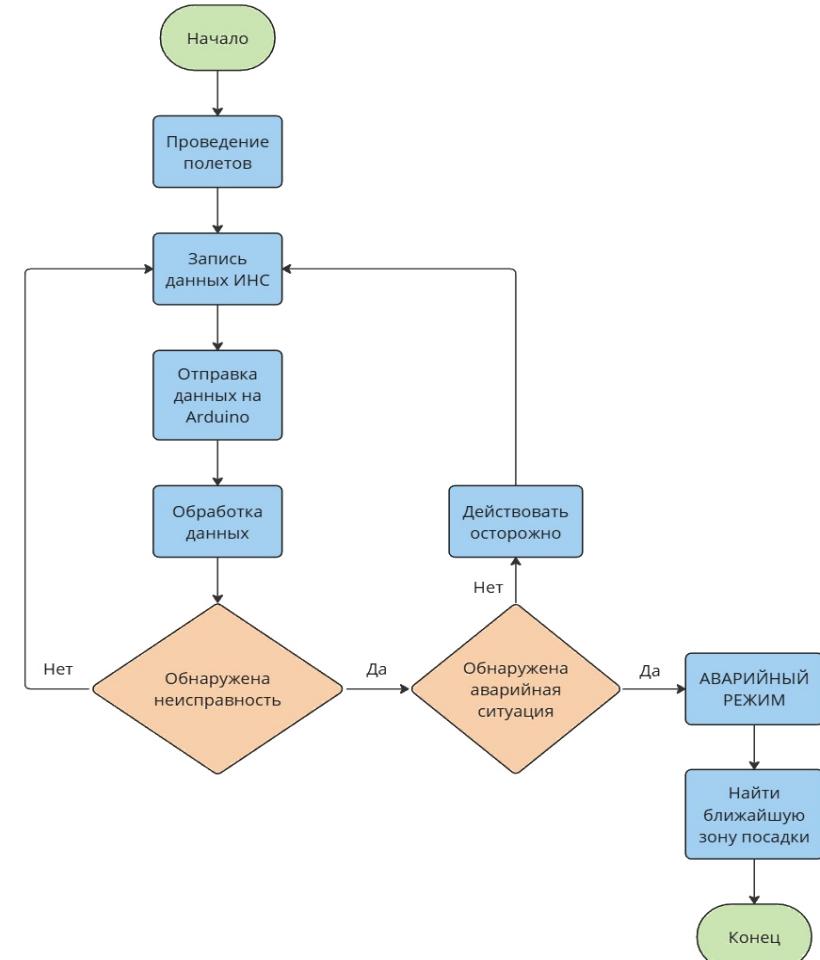
## Действия при аварийной ситуации

### Три этапа:

1. Детектирование АС
2. Выбор места посадки
3. Аварийная посадка

### Почему это важно?

Несвоевременное обнаружение аварийной ситуации может привести к критическим последствиям





## Данные для обучения

БПЛА (Holybro x500)



Поврежденные пропеллеры



100 полётов ~ по 2 минуты

По 20 полетов с  
поврежденными  
пропеллерами в  
количестве 0, 1, 2, 3 и 4 шт

```
{"timestamp":216372882,"gyro_rad":  
[0.0051484676,0.0001790887,-0.0011881116],"gyro_integral_dt":4997,"accelerometer_timestamp_relative":0,"accelerometer_m_s2":  
[-0.0130835138,-0.0602831133,-9.8212537766],"accelerometer_integral_dt":4997,"accelerometer_clipping":0}  
{"timestamp":216373481,"gyro_rad":  
[0.0028427013,0.0109202936,-0.001164366],"gyro_integral_dt":4996,"accelerometer_timestamp_relative":0,"accelerometer_m_s2":  
[-0.0514940731,-0.0104970001,-9.7602043152],"accelerometer_integral_dt":4996,"accelerometer_clipping":0}  
 {"timestamp":216380975,"gyro_rad":  
[0.0070812008,0.0000825309,-0.0001918009],"gyro_integral_dt":4995,"accelerometer_timestamp_relative":0,"accelerometer_m_s2":  
[-0.0010820113,-0.0447425321,-9.8283319473],"accelerometer_integral_dt":4995,"accelerometer_clipping":0}
```



## Загрузка и первичная обработка данных

### Загрузка данных с GitHub

```
[2]: url = "https://github.com/tiiuae/UAV-Realistic-Fault-Dataset/archive/refs/heads/main.zip"
response = requests.get(url)

zip_file = zipfile.ZipFile(io.BytesIO(response.content))
zip_file.extractall("UAV-Realistic-Fault-Dataset")
```

```
[4]: # Убедимся, что данные корректно записались
test_data = pd.read_csv(r"ProcessedDataset\0\data_0.csv")
test_data.head(5)
```

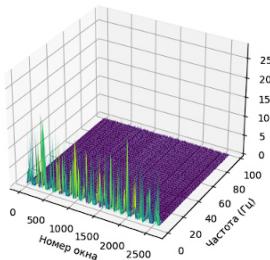
	time	glnt	alnt	gx	gy	gz	ax	ay	az
0	947975062	4998	4998	0.008529	0.004348	-0.001269	0.145165	-0.034807	-9.828311
1	947976312	4996	4996	0.001431	0.005915	-0.001310	0.116024	0.020439	-9.769987
2	947983469	4998	4998	0.016173	0.007350	0.000573	0.092866	0.060234	-9.827328
3	947996546	4994	4994	-0.001588	0.000100	-0.003505	0.065213	0.016342	-9.804899
4	947995456	4995	4995	0.008024	0.012801	0.000042	0.082880	0.064342	-9.800008

Name	Modified	File Size
0	2d ago	
1	14d ago	
2	14d ago	
3	14d ago	
4	14d ago	
class_0_number_0.csv	14 days ago	1.9 MB
class_0_number_1.csv	14 days ago	1.7 MB
class_0_number_10.csv	14 days ago	1.6 MB
class_0_number_11.csv	14 days ago	2 MB
class_0_number_12.csv	14 days ago	1.7 MB
class_0_number_13.csv	14 days ago	1.6 MB
class_0_number_14.csv	14 days ago	1.8 MB
class_0_number_15.csv	14 days ago	1.5 MB
class_0_number_16.csv	14 days ago	1.8 MB
class_0_number_17.csv	14 days ago	1.7 MB
class_0_number_18.csv	14 days ago	1.5 MB
class_0_number_2.csv	14 days ago	1.6 MB
class_0_number_3.csv	14 days ago	1.9 MB
class_0_number_4.csv	14 days ago	1.6 MB
class_0_number_5.csv	14 days ago	1.8 MB
class_0_number_6.csv	14 days ago	1.8 MB
class_0_number_7.csv	14 days ago	1.6 MB
class_0_number_8.csv	14 days ago	1.7 MB
class_0_number_9.csv	14 days ago	2 MB
data_0.csv	14 days ago	32.7 MB

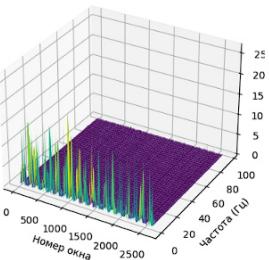


## Построение 3-D спектрограмм

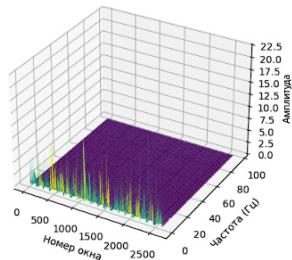
3D Спектрограмма для gx, class 0



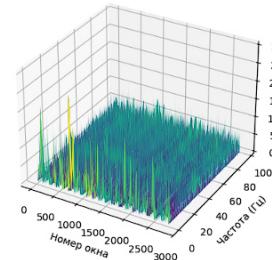
3D Спектрограмма для gy, class 0



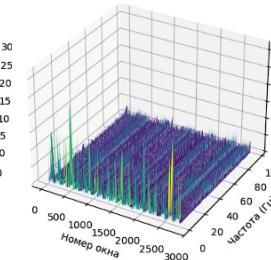
3D Спектрограмма для gz, class 0



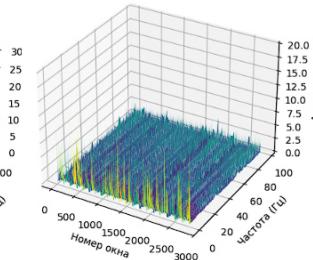
3D Спектрограмма для gx, class 4



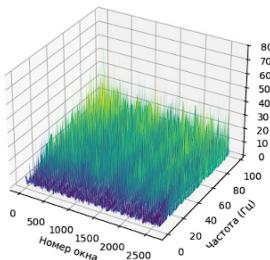
3D Спектрограмма для gy, class 4



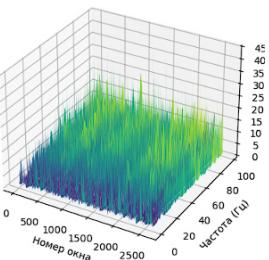
3D Спектрограмма для gz, class 4



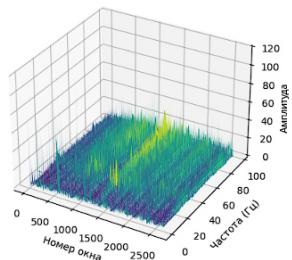
3D Спектрограмма для ax, class 0



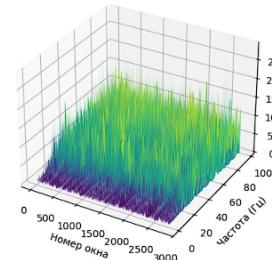
3D Спектрограмма для ay, class 0



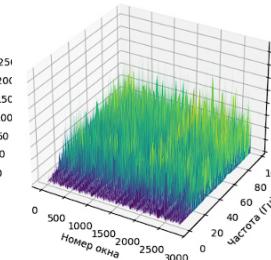
3D Спектрограмма для az, class 0



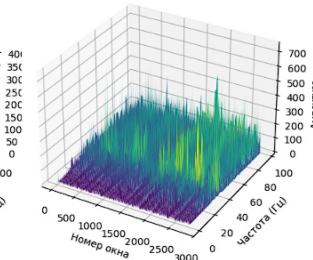
3D Спектрограмма для ax, class 4



3D Спектрограмма для ay, class 4



3D Спектрограмма для az, class 4



Класс «0»

Класс «4»

Можно увидеть заметное увеличение амплитуды для трёх осей акселерометра и появление дефектных частот также для трёх осей гироскопа в диапазоне ~ от 10 Гц.



# Извлечение признаков

```
# ЧО - частотная область, ВО - временная область
cols = ["ЧО Медиана", "ЧО Средняя",
        "ЧО Std", "ЧО Max", "ЧО Min",
        "ЧО 90 процентиль", "ЧО 75 процентиль", "ЧО 25 процентиль",
        "ЧО Куртозис", "ЧО Ассиметрия", "ЧО Энергия", "ЧО Вариация",
        "ЧО Количество пиков",
        "ЧО Средняя > 10 Гц", "ЧО Std > 10 Гц", "ЧО Энергия > 10 Гц",
        "ВО Медиана", "ВО Средняя",
        "ВО Std", "ВО Max", "ВО Min",
        "ВО 90 процентиль", "ВО 75 процентиль", "ВО 25 процентиль",
        "ВО Куртозис", "ВО Ассиметрия", "ВО Энергия", "ВО Вариация",
        "ВО Количество пиков",
        "ВО Общая мощность", "ВО Средняя мощность"]

channels = ["gx", "gy", "gz", "ax", "ay", "az"]
cols = sum([[col[:3] + name + col[2:] for col in cols] for name in channels], [])
cols.append("Класс")
```

## Пространство для будущих исследований:

Провести углубленный анализ влияния каждого признака на точность классификации с целью выявления наиболее значимых без потери точности модели

Суммарно из частотной и временной областей – 186 признаков (по 31 признаку на каждую из осей акселерометра и гироскопа)

```
features = []
for channel in range(amplitude.shape[1]):
    amp_half = amplitude[:,overlap, channel] # Берем только положительные частоты
    seg = segment[:, :, channel]

    # Частотный спектр
    features.append(np.median(amp_half)) # Медиана
    features.append(np.mean(amp_half)) # Средняя амплитуда
    features.append(np.std(amp_half)) # Стандартное отклонение
    features.append(np.max(amp_half)) # Максимальная амплитуда
    features.append(np.min(amp_half)) # Минимальная амплитуда
    features.append(np.percentile(amp_half, 90)) # 90-й процентиль
    features.append(np.percentile(amp_half, 75)) # 75-й процентиль
    features.append(np.percentile(amp_half, 25)) # 25-й процентиль
    features.append(kurtosis(amp_half)) # Куртозис
    features.append(skew(amp_half)) # Ассиметрия
    features.append(np.trapz(amp_half ** 2)) # Энергия

    # Коэффициент вариации
    features.append(np.std(amp_half) / np.mean(amp_half))

    # Количество пиков (выбросов)
    features.append(np.sum(amp_half > np.median(amp_half) + 0.5 * np.std(amp_half)))

    # Исследование частот > 10 Гц
    amp_half_10 = amp_half[more10hz:]
    features.append(np.mean(amp_half_10)) # Средняя амплитуда частот > 10 Гц
    features.append(np.std(amp_half_10)) # Стандартное отклонение частот > 10 Гц
    features.append(np.trapz(amp_half_10 ** 2)) # Энергия частот > 10 Гц

    # Временная область
    features.append(np.median(seg)) # Медиана
    features.append(np.mean(seg)) # Средняя амплитуда
    features.append(np.std(seg)) # Стандартное отклонение
    features.append(np.max(seg)) # Максимальная амплитуда
    features.append(np.min(seg)) # Минимальная амплитуда
    features.append(np.percentile(seg, 90)) # 90-й процентиль
    features.append(np.percentile(seg, 75)) # 75-й процентиль
    features.append(np.percentile(seg, 25)) # 25-й процентиль
    features.append(kurtosis(seg)) # Куртозис
    features.append(skew(seg)) # Ассиметрия
    features.append(np.trapz(seg ** 2)) # Энергия

    # Коэффициент вариации
    features.append(np.std(seg) / np.mean(seg))

    # Количество пиков (выбросов)
    features.append(np.sum(seg > np.median(seg) + 0.5 * np.std(seg)))

    # Плотность мощности
    f, Pxx = Welch(seg, fs=fs, window='hann', nperseg>window_size, noverlap=overlap)
    features.append(np.sum(Pxx[:overlap])) # Общая мощность
    features.append(np.mean(Pxx[:overlap])) # Средняя мощность

# Добавляем метку класса и сохраняем извлеченные признаки
features.append(clas)
features_df.loc[len(features_df)] = features
```



# Построение моделей

```
# Загрузка данных
features_df = pd.read_csv('features.csv')
X = features_df.drop(columns = ["Класс"])
y = features_df["Класс"].values

# Разделение на обучающую и тестовую выборки
random_state = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=random_state)

# Приведение к нормальному распределению (Преобразование Yeo-Johnson)
power_transformer = PowerTransformer(method='yeo-johnson')
X_train = power_transformer.fit_transform(X_train)
X_test = power_transformer.transform(X_test)

# Стандартизация данных
standard_scaler = StandardScaler()
X_train = standard_scaler.fit_transform(X_train)
X_test = standard_scaler.transform(X_test)

# Нормализация данных
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Инициализация моделей
models = {
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=4, leaf_size=1, p=1, metric='manhattan', weights='distance'),
    "Naive Bayes": GaussianNB(var_smoothing=0.1),
    "Support Vector Machine": SVC(C=800, gamma=0.1, class_weight='balanced', tol=0.01, probability=True, random_state=random_state),
    "Logistic Regression": LogisticRegression(C=300, solver='newton-cg', random_state=random_state),
    "Decision Tree": DecisionTreeClassifier(max_depth=30, criterion='entropy', random_state=random_state),
    "Perceptron": MLPClassifier(hidden_layer_sizes=(144, 256, 64, 16, 5), max_iter=400, random_state=random_state),
    "Random Forest": RandomForestClassifier(n_estimators=400, max_depth=20, random_state=random_state),
    "CatBoost": CatBoostClassifier(silent=True, random_state=random_state),
    "AdaBoost": AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=6, random_state=random_state), n_estimators=400, learning_rate=0.1),
    "Gradient Boosting": GradientBoostingClassifier(random_state=random_state),
    "Histogram-based Gradient Boosting": HistGradientBoostingClassifier(random_state=random_state),
    "XGBoost": XGBClassifier(random_state=random_state),
    "LightGBM": LGBMClassifier(verbose=-1, random_state=random_state),

    "Stacking Classifier": StackingClassifier(
        estimators=[("LGBMClassifier", LGBMClassifier(verbose=-1, random_state=random_state)),
                    ("Random Forest", RandomForestClassifier(n_estimators=400, max_depth=20, random_state=random_state)),
                    ("XGBCClassifier", XGBClassifier(random_state=random_state)),
                    ("CatBoost", CatBoostClassifier(silent=True, random_state=random_state)),
                    ("Histogram-based Gradient Boosting", HistGradientBoostingClassifier(random_state=random_state)),
                    ("K-Nearest Neighbors", KNeighborsClassifier(n_neighbors=4, leaf_size=1, p=1, metric='manhattan', weights='distance')),
                    ("Support Vector Machine", SVC(C=800, gamma=0.1, class_weight='balanced', tol=0.01, probability=True, random_state=random_state))],
        final_estimator=LogisticRegression(random_state=random_state)),

    "Voting Classifier": VotingClassifier(
        estimators=[("LGBMClassifier", LGBMClassifier(verbose=-1, random_state=random_state)),
                    ("Random Forest", RandomForestClassifier(n_estimators=400, max_depth=20, random_state=random_state)),
                    ("XGBCClassifier", XGBClassifier(random_state=random_state)),
                    ("CatBoost", CatBoostClassifier(silent=True, random_state=random_state)),
                    ("Histogram-based Gradient Boosting", HistGradientBoostingClassifier(random_state=random_state)),
                    ("K-Nearest Neighbors", KNeighborsClassifier(n_neighbors=4, leaf_size=1, p=1, metric='manhattan', weights='distance')),
                    ("Support Vector Machine", SVC(C=800, gamma=0.1, class_weight='balanced', tol=0.01, probability=True, random_state=random_state)),
                    ("Logistic Regression", LogisticRegression(C=300, solver='newton-cg', random_state=random_state)),
                    ("Perceptron", MLPClassifier(hidden_layer_sizes=(144, 256, 64, 16, 5), max_iter=400, random_state=random_state))],
        voting='soft')
}
```

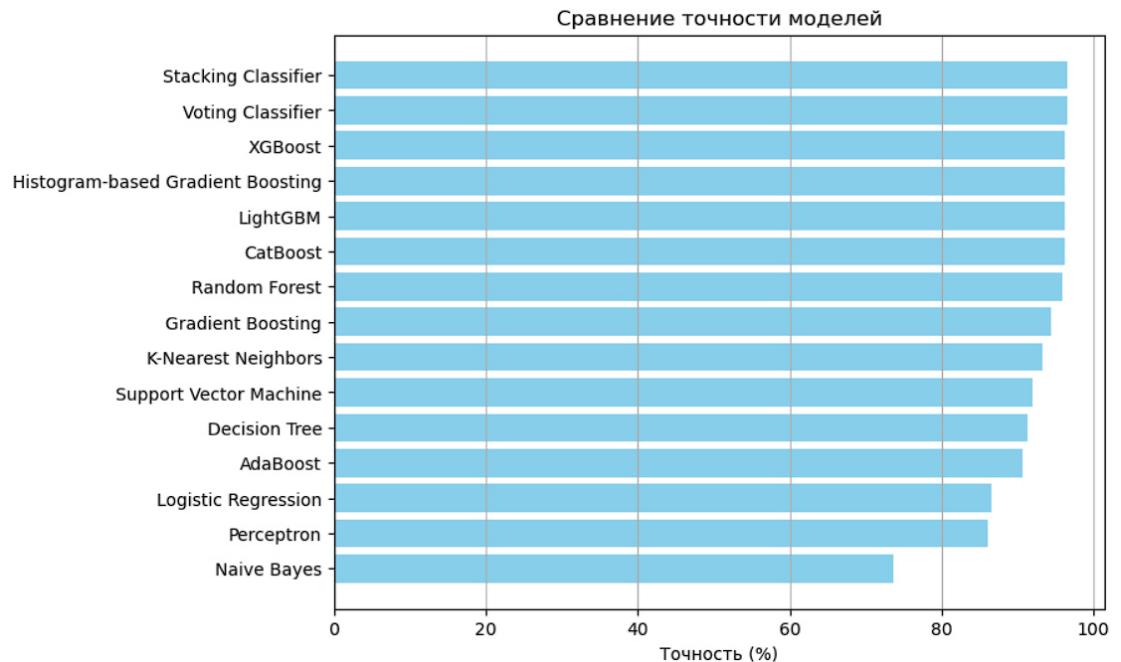
- 15 моделей -



# Сравнение моделей

## Сортировка по столбцу Accuracy

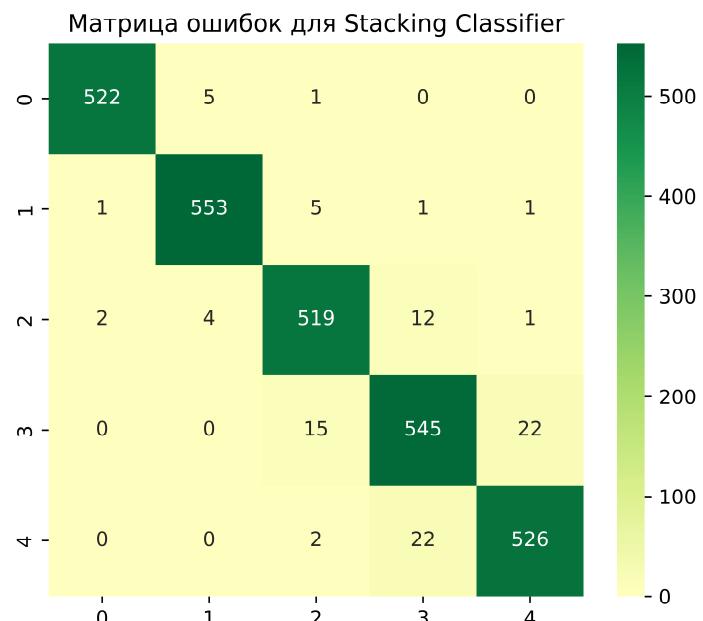
	Accuracy	Precision	Recall	F1 Score
<b>Stacking Classifier</b>	96.593	96.595	96.593	96.593
<b>Voting Classifier</b>	96.484	96.486	96.484	96.484
<b>XGBoost</b>	96.267	96.270	96.267	96.267
<b>Histogram-based Gradient Boosting</b>	96.231	96.229	96.231	96.230
<b>LightGBM</b>	96.194	96.199	96.194	96.197
<b>CatBoost</b>	96.158	96.155	96.158	96.154
<b>Random Forest</b>	95.868	95.857	95.868	95.860
<b>Gradient Boosting</b>	94.346	94.354	94.346	94.347
<b>K-Nearest Neighbors</b>	93.295	93.292	93.295	93.291
<b>Support Vector Machine</b>	91.881	91.874	91.881	91.874
<b>Decision Tree</b>	91.301	91.281	91.301	91.280
<b>AdaBoost</b>	90.721	91.589	90.721	90.696
<b>Logistic Regression</b>	86.589	86.625	86.589	86.592
<b>Perceptron</b>	86.082	87.877	86.082	86.311
<b>Naive Bayes</b>	73.686	76.717	73.686	73.304



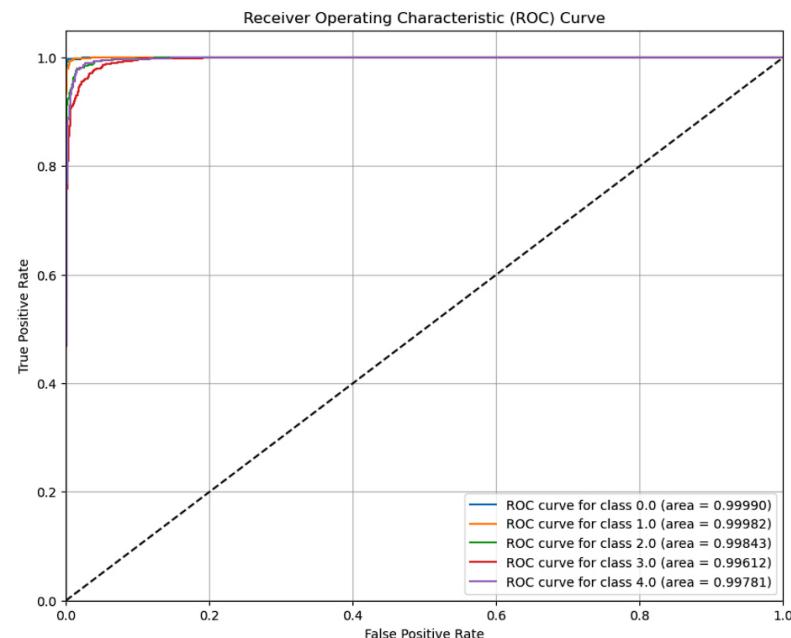
Лучшая модель — Stacking Classifier



## Анализ лучшей модели



	Accuracy	Precision	Recall	F1 Score
Stacking Classifier	96.593	96.595	96.593	96.593



```
# Расчет точности с учетом ошибки в классе не более чем на 1

y_pred = models.get("Stacking Classifier").predict(X_test)

correct_predictions = 0
for true_class, predicted_class in zip(y_test, y_pred):
    if abs(true_class - predicted_class) <= 1:
        correct_predictions += 1

new_accuracy = correct_predictions / len(y_test)
print(f"Точность с учетом ошибки в классе не более чем на 1: {round(new_accuracy * 100, 2)} %")
```

Точность с учетом ошибки в классе не более чем на 1: 99.71 %



# Создание и обучение нейронной сети

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	23,936
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 512)	66,048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131,328
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 64)	16,448
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 5)	325

Total params: 714,257 (2.72 MB)

Trainable params: 238,085 (930.02 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 476,172 (1.82 MB)

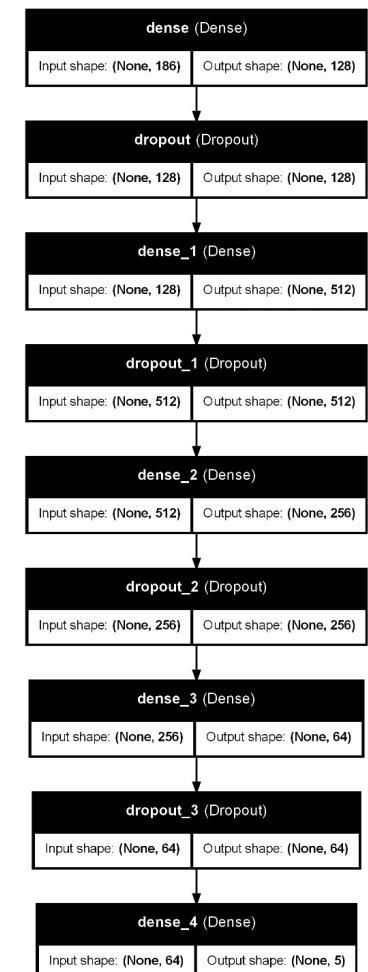
Время обучения ~ 2-3 минуты

Epoch 504/1000

69/69 ━━━━━━ 0s 3ms/step - accuracy: 0.9629 - loss: 0.1017 - val\_accuracy: 0.9162 - val\_loss: 0.2777

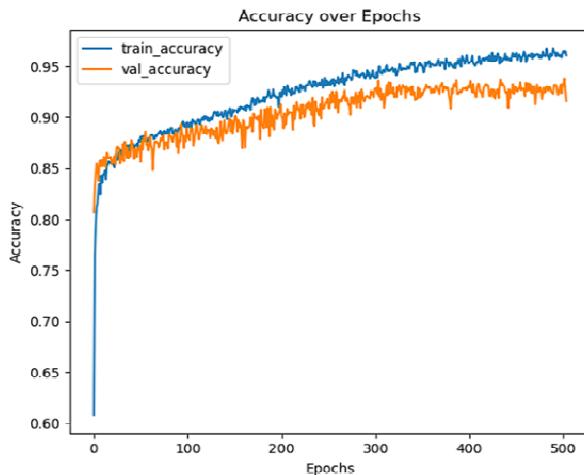
Epoch 504: early stopping

Restoring model weights from the end of the best epoch: 304.



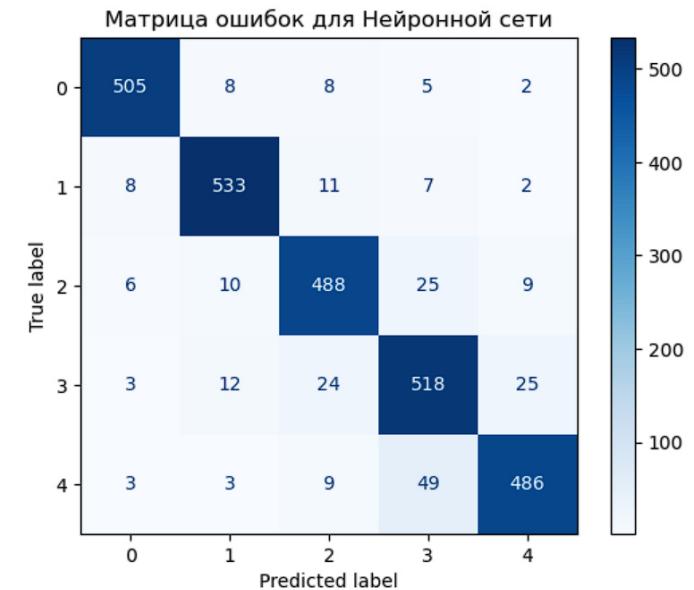
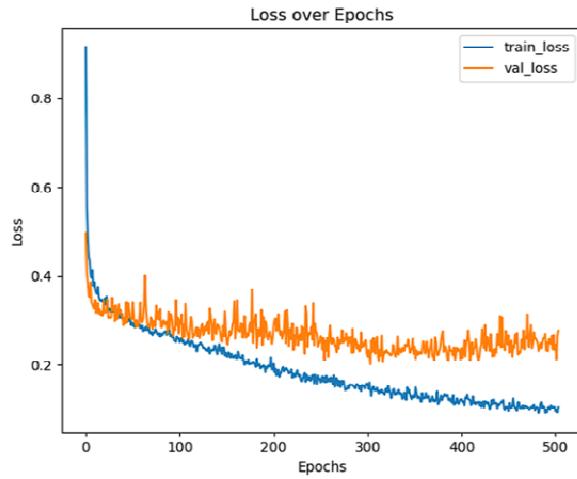


## Оценка точности нейронной сети



	precision	recall	f1-score	support
0	0.96	0.96	0.96	528
1	0.94	0.95	0.95	561
2	0.90	0.91	0.91	538
3	0.86	0.89	0.87	582
4	0.93	0.88	0.91	550
accuracy			0.92	2759
macro avg	0.92	0.92	0.92	2759
weighted avg	0.92	0.92	0.92	2759

Accuracy: 91.7 %  
Precision: 91.758 %  
F1-Score: 91.712 %  
Recall: 91.7 %



Точность ниже, чем у ансамблевых методов, но выше, чем у большинства одиночных моделей



ЦЕНТР  
ДОПОЛНИТЕЛЬНОГО  
ОБРАЗОВАНИЯ  
МГТУ им. Н.Э. Баумана

# Приложение

## Главная страница

The screenshot shows a web browser window with the URL 127.0.0.1:5000. A central modal dialog box is displayed, prompting the user to "Загрузите файл логов полёта в формате JSONL". Below the prompt are two input fields: "Выберите файл" and "Файл не выбран". A blue "Загрузить" button with an upward arrow icon is located at the bottom of the dialog. The background of the page is light gray.

Результат обработки

## Предсказанные классы:

0
0
0
0
0
0
0
0
0
0
1
0
0
0
0
0

[Загрузить другой файл](#)



ЦЕНТР  
ДОПОЛНИТЕЛЬНОГО  
ОБРАЗОВАНИЯ  
МГТУ им. Н.Э. Баумана

## Приложение

### Предсказанные классы:

Ошибка: Неверный файл

[Загрузить другой файл](#)

Загружен некорректный  
файл JSONL

Пожалуйста, загрузите  
файл формата JSONL

[Вернуться к загрузке файла](#)

Загружен не JSONL файл



## Перспективы дальнейших исследований

- Корректировка гиперпараметров моделей
- Исследование наиболее значимых признаков
- Разработка более эффективных методов детектирования АС
- **Реализация детектирования АС в системе реального времени**
- Использование других датчиков для детектирования АС
- **Создание виртуальной среды для симуляции работы БПЛА.** Цель – моделирование различных сценариев неисправностей, создание синтетических данных и проработка необычных или опасных ситуаций с минимальными затратами ресурсов



ЦЕНТР  
ДОПОЛНИТЕЛЬНОГО  
ОБРАЗОВАНИЯ  
МГТУ им. Н.Э. Баумана

Конец

Спасибо за  
внимание!





ЦЕНТР  
ДОПОЛНИТЕЛЬНОГО  
ОБРАЗОВАНИЯ

МГТУ им. Н.Э. Баумана



[do.bmstu.ru](http://do.bmstu.ru)