

```
import pandas as pd
import numpy as np
import matplotlib

matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from datetime import datetime
import warnings

warnings.filterwarnings('ignore')
```

هر دسته داده‌ای جزوی (CSV, DataFrame) \Leftarrow Pandas

محاسبات عددی، آرایه‌ها، نویه، سینووس، تصادفی \Leftarrow numpy

Backend Selection ensures GUI compatibility across envs



رسم بودار \Leftarrow matplotlib

مخفن درن \Leftarrow TkAgg
برای نمایش

در اینی استفاده کنند مخصوصاً در PyCharm و ویندوز جزوی

بلاک‌های رنگی او ی لند.

← جای رسم بودار plt

```
1 from sklearn.model_selection import train_test_split  
2 from sklearn.linear_model import LinearRegression  
3 from sklearn.preprocessing import StandardScaler  
4 from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error  
5 from datetime import datetime  
import warnings  
  
warnings.filterwarnings('ignore')
```

1 دیتا را به test , train تقسیم کنید.

|
حدبی (Overfitting) ایجاد می‌کند

استاددار ML پیاده‌سازی شده:

ترهیب استاددار ML :

- جمع آوری دیتا

- تغییر حالت دیتا

feature engineering -3

train / test ~ Split -4

5. نمونه‌سازی

6. آموزش مدل

7. ارزیابی (ویسٹ)

8. deploy (احمیت‌ری)

داده‌آوری: خلاصه هفتاد کاری داده‌ها و ترسیم نمودن داده‌های جدید.

ML Model liner model 2

خونی لذت را بفرمایی بین درودی ها و حزبی خنی است.

3 ذمائل سازی و میکارها:
 $\theta_{\text{مبادر}} = 0$

انحراف معیار = ۱

بهای مدل های خنی و CNNX هیلی محض است.

4 هسته ها
→ خطراسته جریمه ای لذت.

→ میانگین خطی داقعی

R^2 → لیفنت مدل (نحو بقدر داده ای داده را توانیم)
داده ای داده را توانیم

خطا = معنادار داقعی - معنادار پیش بینی شده

$MSE \leftarrow \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ به توانی ۲ را از این داده های بالای خطاهای بزرگ را

حیلی هنی بزرگتر نمایش دهد.

بهای هتل لذت خطا \hat{y}_{15} باشد $\Rightarrow MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ را ۱۰۰ نسخه را دهد

خطای بزرگ اندلس عینعادی دیده بیشود پس به افقط لوح جریمه
بیشود.

(JAN 1 1970) تاریخ همار از سال datetime → timestamp ۵
بعترین بای لات و گزارش

→ حذفی تبعی (بای ادانه دانسته عرفان) → warnings.ignore

... > Warning بای موردی معنی بحتر است

هیچ وقت ignore نماید.

```
try:  
    from skl2onnx import convert_sklearn  
    from skl2onnx.common.data_types import FloatTensorType  
    import onnxruntime as ort  
  
    ONNX_AVAILABLE = True  
except ImportError:  
    print("Warning: ONNX libraries not installed. Install with: "  
          "pip install skl2onnx onnxruntime")  
    ONNX_AVAILABLE = False
```

deploy-ready

ONNX ⊂ Import بخش

تلاشی بای sklearn ← ONNX : Import

و اگر ONNX یعنی lib میام با خطای ImportError روبرو شود

آن lib را فیل کن.

دارد یعنی اگر فیل کن جدون خطای ادامه بدهد.

```

class IoTDataPipeline:
    """
    Main class for IoT sensor data pipeline using OOP principles
    """

    def __init__(self, project_name="IoT_Sensor_Project"):
        self.project_name = project_name
        self.raw_data = None
        self.clean_data = None
        self.model = None
        self.scaler = StandardScaler()
        self.X_train = None
        self.X_test = None
        self.y_train = None
        self.y_test = None
        self.predictions = None
        self.onnx_model_path = "temperature_model.onnx"
        self.data_source = None

        print(f"==== {self.project_name} Initialized ===")
        print(f"Timestamp: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n")

```

لیاس اسی (OOP pipeline)

الگوریتم حیل abstraction نویسندگان است.

هر مرحله به دل می

Self.raw_data دیتای خام

Self.clean_data دیتای پیش‌سازی شده

Self.model = None مدل پیش‌سازی شود.

Self.scaler = StandardScaler()

برای ذره‌ای سازی یا استاندارد سازی داده‌ها Scikit-learn استفاده می‌کنیم.

: __init__ در

- تنظیمات اولیه

- مرحله State

- لایه شروع

داده‌ها را به لغواری تغییر دهند می‌انجامیم و انحراف معیار سازی را بخواهیم.

`self.onnx_model_path = "temp.."`

deployable مدلی سبک دارویی
model

```
self.X_train = None
self.X_test = None
self.y_train = None
self.y_test = None
```

حالت های آموزشی State -

- بعد از برای ارزیابی
استفاده کوچک است.

```
# ===== STEP 1: Data Acquisition =====
def generate_synthetic_data(self, n_samples=500, save_csv=True):
    """
    """
    print("=" * 50)
    print("STEP 1: DATA ACQUISITION (SYNTHETIC DATA)")
    print("=" * 50)

1 time = pd.date_range("2025-01-01", periods=n_samples, freq="H")

    hours = np.arange(n_samples)
2 temperature = 20 + 8 * np.sin(2 * np.pi * hours / 24) + np.random.normal(0, 1.5, n_samples)

3 humidity = 60 - 0.5 * temperature + 10 * np.cos(2 * np.pi * hours / 24) + np.random.normal(0, 3, n_samples)
    humidity = np.clip(humidity, 20, 100) # Keep realistic range

4 pressure = 1013 + 5 * np.sin(2 * np.pi * hours / 168) + np.random.normal(0, 2, n_samples)

5 self.raw_data = pd.DataFrame({
        "Timestamp": time,
        "Temperature": temperature,
        "Humidity": humidity,
        "Pressure": pressure
    })

6 missing_indices = np.random.choice(n_samples, size=int(0.05 * n_samples), replace=False)
    self.raw_data.loc[missing_indices, 'Temperature'] = np.nan
```

تولید دیتای معنی‌داری :

1 ساعتی timestamp تولید

time series پایانی

تاریخ (تعداد)
 \uparrow
 \downarrow
 \downarrow
 \downarrow
 np.random.normal(0, 1.5, 3)

متغیرهای مانند
 اندک افت و معیار

± الگوی روزانه (Sin) می‌سینوسی برای دادا

و اضافه دردن noise با اعداد رندوم

IOT مبتنی بر داده‌های دنیای واقعی

`h = np.array([10, 50, 120])`

`np.clip(h, 12, 100)`

↓

↳

در آرایه هر عدد
در آرایه هر عددی دو حلقه
از 12 بود و 12 تا 100
بین
بیند.

برگردان از 100 به 100
تبديل جسمید.

humidity 3

جیستی مخلوقات
جا داشت

لسانی همی نه واقعیت داریم.

[12, 50, 100]

نهایی تمامی زیج
کی مادن.

pressure 4

1013 + Slow Sin + noise

تفصیلات

realistic atmospheric pressure

5 جبری همچنین DataFrame استاندارد

`self.raw_data = pd.DataFrame({...})`

DataFrame ی داتا فریم در Pandas ن جدول داده ای یعنی \Rightarrow درستی کند.

هر لایه دیسکریتی اسم مستون یا سُور

معادل ماس لیست یا آرایه داده های مستون یا سُور.

result

ن عنوان هنال این قسمت

از ن ب این قروات در

ک این

	Timestamp	Temperature	Humidity	Pressure
0	2025-01-01 00:00:00	22.055317	56.580904	1014.676995
1	2025-01-01 01:00:00	22.280293	57.992828	1012.981653
2	2025-01-01 02:00:00	24.380085	58.040747	1011.619750
3	2025-01-01 03:00:00	25.451824	58.986332	1011.449707
4	2025-01-01 04:00:00	24.636286	52.259963	1012.117358

missing values 6

سُلْطَنِي سازی حزبِ ایامِ مسیح

اِن حَسْبَتْ بِهِي مَرْحَمَى Cleaning نِسْكَتْ بِنَازِ اللَّهِ.

```
missing_indices = np.random.choice(n_samples, size=int(0.05 * n_samples), replace=False)
self.raw_data.loc[missing_indices, 'Temperature'] = np.nan
```

4

با خود از دینای Temp

نے نام تبدیل کر دیا

١٥. از لذت نوی انتخابی استون.

يُلْعَنُ مَنْ لَا يَأْتِي بِالْحَقَّ

حروفی این فہمت:

	Timestamp	Temperature	Humidity	Pressure
0	2025-01-01 00:00:00	NaN	57.922378	1011.019956
1	2025-01-01 01:00:00	22.073314	61.559738	1013.530949
2	2025-01-01 02:00:00	NaN	58.885357	1015.475972
3	2025-01-01 03:00:00	24.625132	51.888764	1015.010096
4	2025-01-01 04:00:00	28.985878	49.418597	1013.728790

یا التفاب حارج دیتا هی تواند از kaggle هم وارد شود:

```
def import_real_data(self, save_csv=True):...
```

```

# ====== STEP 2: DATA CLEANING & PREPROCESSING ======
def clean_and_preprocess(self):
    """
    """
    print("=" * 50)
    print("STEP 2: DATA CLEANING & PREPROCESSING")
    print("=" * 50)

    if self.raw_data is None:
        raise ValueError("No raw data available. Run generate_synthetic_data() or import_real_data() first.")

    self.clean_data = self.raw_data.copy()

    # 1. هنگام قصعه مای کم شده
    missing_before = self.clean_data['Temperature'].isna().sum()
    self.clean_data['Temperature'].fillna(method='ffill', inplace=True)
    self.clean_data['Temperature'].fillna(method='bfill', inplace=True)
    print(f"\u2713 Filled {missing_before} missing temperature values")

    # 2. Remove outliers using IQR method
    Q1 = self.clean_data['Temperature'].quantile(0.25)
    Q3 = self.clean_data['Temperature'].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers_before = len(self.clean_data)
    self.clean_data = self.clean_data[
        (self.clean_data['Temperature'] >= lower_bound) &
        (self.clean_data['Temperature'] <= upper_bound)
    ]
    outliers_removed = outliers_before - len(self.clean_data)
    print(f"\u2713 Removed {outliers_removed} outliers (IQR method)")

```

Cleaning and preprocessing

۱. NaN بود از قبلی جایزه‌اند.
 $\text{NaN} \leftarrow \text{ffill}$

۲. NaN بود از پسی باید پنهان شود.
 $\text{NaN} \leftarrow \text{bfill}$

inplace اجرا نمود.

جدید یا لیست درست نمود.

2 استفاده از روش آماری استاندارد IQR

$$IQR = Q_3 - Q_1$$

فاصله‌ای‌ین چارت‌سوم، چارت‌اول
حذف \times حینی بالاترها

داده‌های خارج از این چارت

$$Q_3 + 1,5^* IQR \rightarrow$$

$$Q_1 - 1,5^* IQR \leftarrow$$

داده‌های از این پایین در \times حذف

Outlier ناخنجرد

حسابی سواد.

```

# 3. Apply moving average smoothing
window_size = 5
self.clean_data['Temperature_Smooth'] = self.clean_data['Temperature'].rolling(
    window=window_size, center=True
).mean()
self.clean_data['Temperature_Smooth'].fillna(
    self.clean_data['Temperature'], inplace=True
)
print(f"\u2708 Applied moving average filter (window={window_size})")

# 4. Feature engineering: create time-based features
self.clean_data['Timestamp'] = pd.to_datetime(
    self.clean_data['Timestamp'],
    utc=True,
    errors='coerce'
)
2 self.clean_data['Hour'] = pd.to_datetime(self.clean_data['Timestamp']).dt.hour
self.clean_data['Day'] = pd.to_datetime(self.clean_data['Timestamp']).dt.day
print(f"\u2708 Created time-based features")

print(f"\nCleaned dataset shape: {self.clean_data.shape}")
print("Statistics after cleaning:")
print(self.clean_data[['Temperature', 'Humidity', 'Pressure']].describe())
print()

return self.clean_data

```

rolling (window=5, center=True).mean() ۱

حذف noise

حذف trend

: آرایه میانگین سیگنال Signal processing ۲

Temperature	Index
20	0
22	1
24	2
23	3
21	4

دواردهنگ ۳ \Leftarrow window ۳
Center = True

نحوه محاسبه میانگین Index ۰
Nan

Temperature_Smooth	Temperature	Index
20	20	0
22	22	1
23	24	2
22.67	23	3
21	21	4

$$\frac{24 + 22 + 20}{3} = 22 \Leftarrow \text{Index 1}$$

$$\frac{22 + 24 + 23}{3} = 23 \Leftarrow \text{Index 2}$$

$$\frac{24 + 23 + 21}{3} = 22.67 \Leftarrow \text{Index 3}$$

نحوه محاسبه میانگین Index 4
Nan

ویدئوهای زمانی برای آنست الگوهای رفتاری روزانه استخراج شدند.

Feature Engineering ۲

زمان \rightarrow ویدئی عددی

ML-friendly

Temporal Features were extracted

to capture daily behavioral patterns.

هر دیگری Timestamp نیز date و time بعدهاں ہیں ہے

Pandas استادزارد

ال مقادیر خراپ جائے و بدلن لسے ی ہے۔ Nat نے

```
def train_model_and_export_onnx(self, target='Temperature'):
    """
    ...
    print("=" * 50)
    print("STEP 3: MODEL TRAINING & ONNX EXPORT")
    print("=" * 50)

    if self.clean_data is None:
        raise ValueError("No clean data available. Run clean_and_preprocess() first.")

    feature_cols = ['Humidity', 'Pressure', 'Hour']
    X = self.clean_data[feature_cols].values
    y = self.clean_data[target].values
```

2 self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(

X, y, test_size=0.2, random_state=42, shuffle=True)

دسترو چن اچہ رندم جنڈارد ہے ایک ایسا نیوں کا دلیل ہے

نیوں کا دلیل ہے 42 یہ ہواں واحدی دھر۔
اگر بخواہیم جواب دیئے کہ عدد را عومن لئیں

Training and ONNX export

feature_cols = [...] =>

1- درودی ہائی مارٹ

Target =>

Temperature

2- سروچ یادی مارل جا دیتی موردنظر (اویا لڈ تو ولیع) (حصہ۔

```

self.X_train_scaled = self.scaler.fit_transform(self.X_train)
self.X_test_scaled = self.scaler.transform(self.X_test)
print(f"✓ Applied StandardScaler normalization")

self.model = LinearRegression()
self.model.fit(self.X_train_scaled, self.y_train)

train_score = self.model.score(self.X_train_scaled, self.y_train)
print(f"✓ Model trained successfully")
print(f" Training R² Score: {train_score:.4f}")

if ONNX_AVAILABLE:
    try:
        initial_type = [('float_input', FloatTensorType([None, X.shape[1]]))]
        onnx_model = convert_sklearn(
            self.model,
            initial_types=initial_type,
            target_opset=12
        )

        with open(self.onnx_model_path, "wb") as f:
            f.write(onnx_model.SerializeToString())

        print(f"✓ Model exported to ONNX: {self.onnx_model_path}")
    except Exception as e:
        print(f"✗ ONNX export failed: {e}")
else:
    print("✗ ONNX export skipped (libraries not available)")

print()
return self.model

```

SelfScaler.fit_transform(X-train)

جھٹکا ماری Train fit Scaler

data leakage جھٹکا اور

Self.model = LinearRegression()

Self.model.fit(...)

آموزشی مدار

در این قسمت از دوره:

Same Model - different runtime

جیود دارد.

ONNX_AVAILABLE \Rightarrow T بعد ممکن است ONNX lib یعنی اگر

این کد وارد if نمی شود.

onnxruntime \Leftarrow تا ONNX

Sklearn fallback \Leftarrow تا ONNX

Self.predictions = ort_session.run(...)

```
# ===== STEP 5: EVALUATION & VISUALIZATION =====
def evaluate_and_visualize(self):
    """
    Evaluate model performance and create visualizations
    """
    print("=" * 50)
    print("STEP 5: EVALUATION & VISUALIZATION")
    print("=" * 50)

    if self.predictions is None:
        raise ValueError("No predictions available. Run load_onnx_and_predict() first.")

    # Calculate metrics
    mse = mean_squared_error(self.y_test, self.predictions)
    mae = mean_absolute_error(self.y_test, self.predictions)
    r2 = r2_score(self.y_test, self.predictions)
    rmse = np.sqrt(mse)

    print("Model Performance Metrics:")
    print(f" - Mean Squared Error (MSE): {mse:.4f}")
    print(f" - Root Mean Squared Error (RMSE): {rmse:.4f}")
    print(f" - Mean Absolute Error (MAE): {mae:.4f}")
    print(f" - R2 Score: {r2:.4f}")
    print()

    # Create visualizations
    self._create_visualizations()

    return {
        'mse': mse,
        'rmse': rmse,
        'mae': mae,
        'r2': r2
    }
```

Step 5 Evaluation and Visualization

ML-report

نمایش اسناد

MSE

RMSE

MAE

R²

مترب
ها:

Visualization Dashboard

Actual vs Predict _1

: نمودار

Residual Plot _2

Error distribution _3

Time-Series Smooth VS raw _4

Correlation heatmap _5

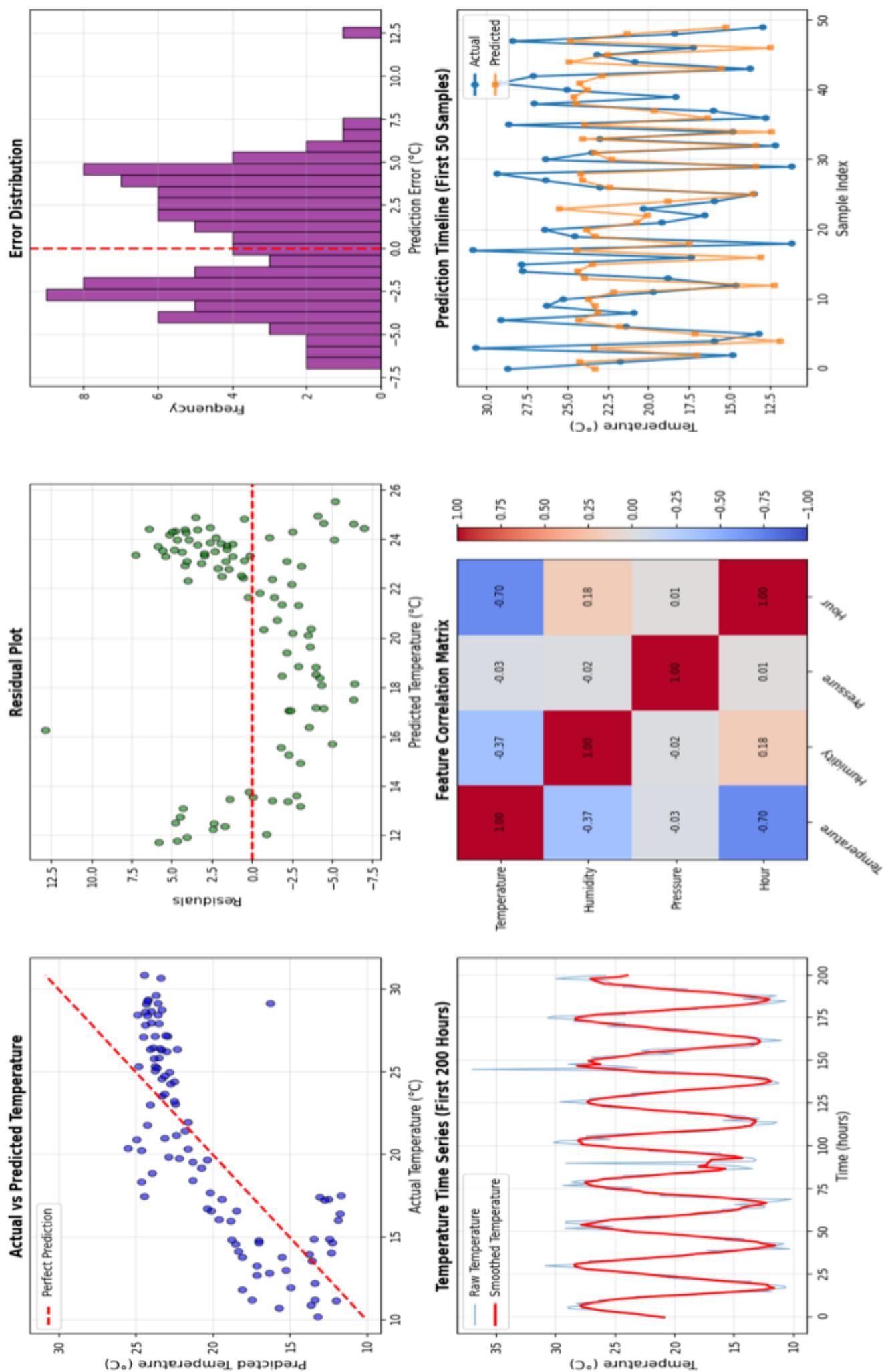
Prediction Time line _6

```
💡 def _create_visualizations(self): ...
```

و در معنی بدلی خوبی را مسأله هدی کنی

ن این قی خودار توانیم سئوه الذا:

Temperature Prediction System - Model Evaluation Dashboard



```
def generate_report(self, metrics):
    """
    print("\n" + "=" * 50)
    print("FINAL PROJECT REPORT")
    print("=" * 50)

    data_source_info = "Synthetic (Generated)" if self.data_source == "synthetic" else "Real (Kaggle Weather Dataset)"

    report = f"""...
    print(report)

    # Save report to file
    with open('project_report.txt', 'w', encoding='utf-8') as f:
        f.write(report)

    print("✓ Report saved to: project_report.txt")
    print("=" * 50)
    print("\n PROJECT COMPLETED SUCCESSFULLY! \n")
```

خلاقانهی pipeline

Kaggle ساختنی داده ← dataSource

متربیت‌ها

نتیجہ‌لندی‌ها

نتایج‌ها با عنوان آدرس نهاده هستند در فایل .txt. دستگاهی می‌شود.

فکری است: Orchestrator main() هاست.

generate از طریق آن کارگزاری خود را داده که kaggle از input

و کارکرد انجام نتایج مراحل را خواهد داشت.

```
def main():
    """..."""
    # Initialize pipeline
    pipeline = IoTDataPipeline(project_name="Temperature_Prediction_System")

    # Step 1: Choose data source
    print("Select data source:")
    print("1. Generate synthetic sensor data")
    print("2. Import real data from Kaggle")

    choose_generate_model = input("\nEnter your choice (1 or 2): ").strip()

    if choose_generate_model == "1":
        pipeline.generate_synthetic_data(n_samples=500, save_csv=True)
    elif choose_generate_model == "2":
        pipeline.import_real_data()
    else:
        print("Invalid choice. Defaulting to synthetic data generation.")
        pipeline.generate_synthetic_data(n_samples=500, save_csv=True)

    # Step 2: Clean and preprocess data
    pipeline.clean_and_preprocess()

    # Step 3: Train model and export to ONNX
    pipeline.train_model_and_export_onnx(target='Temperature')

    # Step 4: Load ONNX model and predict
    pipeline.load_onnx_and_predict()

    # Step 5: Evaluate and visualize results
    metrics = pipeline.evaluate_and_visualize()

    # Generate final report
    pipeline.generate_report(metrics)
```

١٨ "اَهْمَامِ دَالِيَّوْنَتْ"

دُعَيْنِ سَهْبَارِي رَاد

"KOFKOFIKA"