Этот датасет — типичный пример промышленной диагностики оборудования (скорее всего, это **технологическая установка с тепловыми и вибрационными датчиками**, где система отслеживает отклонения и определяет тип неисправности).
Ты уже привела все признаки — давай разберём их **по смысловым группам** и **по физике процесса**.

🧠 Общая идея датасета

Это *мультисенсорная система мониторинга*, которая измеряет:

- **основные физические параметры процесса** (температура, вибрация, давление, поток, ток, напряжение),
- **их спектральные характеристики (через FFT)** — то есть частотное поведение сигналов, важное для поиска аномалий и резонансов,
- и **метку неисправности (`Fault_Type`)**, указывающую, что произошло.

Таким образом, признаки делятся на **два типа**:

1. **временные сигналы в физическом пространстве**,
2. **их частотные компоненты (через преобразование Фурье — FFT)**, где можно увидеть закономерности, не видимые напрямую во времени.

🔹 Основные физические признаки

Признак	Что измеряет	Физический смысл
`Temperature`	Температура системы или узла	Показывает тепловое состояние. Рост температуры может сигнализировать о трении, перегрузке, нарушении теплоотвода.
`Vibration`	Уровень вибрации (амплитуда, RMS, или ускорение)	Главный индикатор механического дефекта: дисбаланс, износ подшипника, несоосность, резонанс.
`Pressure`	Давление в системе (жидкости или газа)	Отражает устойчивость потока, наличие засоров, утечек, перепадов.
`Flow_Rate`	Скорость или объёмный расход среды	Важен для оценки циркуляции жидкости/газа; падение расхода может указывать на закупорку.
`Current`	Ток (A)	Электрическая нагрузка на двигатель, насос или реактор. Рост тока = возросшая нагрузка или трение.
`Voltage`	Напряжение (V)	Питание оборудования. Падение напряжения = сбой в сети; рост = возможная неисправность регулирования.

🔸 Спектральные признаки (FFT)

Вот где начинается интересное:
`FFT_Temp_0 … FFT_Temp_9`, `FFT_Vib_0 … FFT_Vib_9`, `FFT_Pres_0 … FFT_Pres_9` — это **амплитуды спектра частотных компонент** сигналов температуры, вибрации и давления.

📈 Что означает «FFT_X_i»

- **FFT** — *Fast Fourier Transform* (быстрое преобразование Фурье), то есть разложение сигнала по частотам.
- **i = 0…9** — индекс частотного диапазона или бина спектра (например, низкие частоты → i=0, высокие → i=9).
- Значение — **энергия (амплитуда) компоненты сигнала** в данном частотном диапазоне.

📊 Примеры:

Признак	Описание
`FFT_Temp_0`	Энергия низкочастотных колебаний температуры (медленные тренды, инерционность системы).
`FFT_Temp_9`	Энергия высокочастотных колебаний температуры (резкие колебания — сбои термоконтроля, импульсы).
`FFT_Vib_0`	Низкочастотные вибрации — возможный дисбаланс, расцентровка.
`FFT_Vib_9`	Высокочастотные вибрации — износ подшипников, микротрещины, резонанс.
`FFT_Pres_0`	Медленные колебания давления (регулирование, циклы).
`FFT_Pres_9`	Быстрые пульсации давления — турбулентность, кавитация, клапанные удары.

🧩 Таким образом, **каждая группа FFT-признаков отражает «спектр поведения»** определённого параметра, а не его текущее значение.

🔸 Целевая переменная

Признак	Тип	Смысл
`Fault_Type`	категориальный (int или str)	Класс неисправности (например, `0` = норма, `1` = перегрев, `2` = вибрация, `3` = утечка и т.д.).

На Kaggle у этого датасета указано, что `Fault_Type` принимает значения:

- `0` — отсутствие неисправности (нормальная работа)
- `1` — неисправность, связанная с перегревом
- `2` — неисправность, связанная с утечкой
- `3` — неисправность, связанная с колебаниями напряжения

(это логично с учётом названий параметров — каждая группа соответствует одному типу неисправности).

⚙️ Иерархия признаков (как их удобно структурировать)

```
Physical signals
├── Temperature
├── Vibration
├── Pressure
├── Flow_Rate
├── Current
└── Voltage

Spectral components (FFT)
├── Temperature spectrum: FFT_Temp_0 … FFT_Temp_9
├── Vibration spectrum: FFT_Vib_0 … FFT_Vib_9
└── Pressure spectrum: FFT_Pres_0 … FFT_Pres_9

Target
└── Fault_Type
```

💡 Как это читать в терминах диагностики

- **Temperature** + **FFT_Temp_i** → тепловое состояние системы.
  Если растут высокочастотные компоненты — терморегуляция нестабильна.
- **Vibration** + **FFT_Vib_i** → механическое состояние.
  Пики на частотах `FFT_Vib_6–9` могут указывать на износ подшипников.
- **Pressure** + **FFT_Pres_i** → устойчивость потока.
  Колебания на средних частотах (i≈4–6) = пульсации, возможно — кавитация.
- **Current / Voltage** → электрическая нагрузка.
  Рост тока без роста нагрузки — перегрев, залипание вала, короткое замыкание.
- **Flow_Rate** → физическая доставка среды.
  Падение расхода при нормальном давлении — засор или частичная блокировка.

Я подробно опишу, какие визуализации и метрики нужно строить, почему каждая полезна, как интерпретировать результат, и дам готовый Python-скрипт (с комментариями), который строит:

1. кластерную heatmap-карту корреляций между всеми признаками (Physical + FFT),
2. ранжирование признаков по взаимной информации (mutual information) с `Fault_Type`,
3. корреляции «Physical ↔ соответствующие FFT-бины» (чтобы понять, какие FFT-бины отражают энергетику основного сигнала),
4. дендрограмму / кластеризацию признаков,
5. PCA-загрузки и парные графики по лучшим признакам,
6. (опционально) важности признаков через RandomForest (перестановочная важность рекомендуется позже).

Ниже — сначала краткая методология и интерпретация, затем полный код, который запускается “из коробки”.

Что мы делаем и зачем (коротко)

- **Корреляция (Pearson)** — показывает линейные связи. Полезно, чтобы увидеть, какие признаки дублируют информацию (высокая |r| → можно агрегировать).
- **Hierarchical clustering + heatmap** — структурирует матрицу корреляций, помогает визуально выделить группы признаков (кластеры).
- **Mutual Information (MI)** между признаками и `Fault_Type` — оценивает *любой* (включая нелинейный) связный вклад признака в предсказание класса; даёт ранжирование по информативности.
- **Physical ↔ FFT** корреляции — ответят на вопрос «какие FFT-бины лучше всего отражают значение физического сигнала (например, Vibration)».
- **PCA / Loadings** — покажут, какие признаки формируют главные направления изменчивости; полезно для снижения размерности и понимания сочетаний признаков.
- **Pairplots по топ-признакам** — визуально показать, как классы Fault_Type разделяются в пространстве признаков.
- **RandomForest importance / Permutation** — даст эмпирическую важность с учётом нелинейных взаимодействий.

Инструкция по интерпретации (что смотреть)

- В heatmap искать квадраты/блоки с |r| > 0.7 — это сильная взаимосвязь. Такие признаки можно сгруппировать или убрать при построении лёгкой модели.
- Если `Vibration` сильно коррелирует с `FFT_Vib_3..5`, значит средние частоты — ключ к вибрационным дефектам.
- MI: признаки с наибольшим MI — первый кандидат для включения в классификатор. Но MI не даёт направление эффекта — смотри boxplots / pairplots.
- Если много признаков взаимно коррелируют и все важны по MI — вероятна мультиколлинеарность; используем PCA или регуляризацию (ElasticNet).
- С помощью дендрограммы выбери 3–5 кластеров признаков; затем создавай агрегированные признаки (суммы энергии по кластеру, spectral centroid и т.д.).

Готовый код — запусти локально (или дай csv, и я запущу)

Сохранить как `feature_analysis_industrial.py`. Код использует pandas, numpy, matplotlib, seaborn, scipy, sklearn.

```python
# feature_analysis_industrial.py
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.cluster.hierarchy import linkage, dendrogram, leaves_list
from scipy.stats import pearsonr
from sklearn.feature_selection import mutual_info_classif, f_classif
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance

sns.set(style="whitegrid")

def load_data(path):
    df = pd.read_csv(path)
    # ensure Fault_Type is present
    if 'Fault_Type' not in df.columns:
        raise ValueError("CSV must contain 'Fault_Type' column")
    return df

def compute_corr_matrix(df, cols=None, method='pearson'):
    if cols is None:
        cols = [c for c in df.columns if c != 'Fault_Type']
    corr = df[cols].corr(method=method)
    return corr

def plot_clustered_corr(corr, title='Clustered correlation heatmap', outpath=None):
    # distance for clustering: 1 - |corr|
    dist = 1 - np.abs(corr.values)
    # hierarchical clustering
    link = linkage(dist, method='average')
    order = leaves_list(link)
    ordered_cols = corr.columns[order]
    corr_ord = corr.loc[ordered_cols, ordered_cols]

    plt.figure(figsize=(12,10))
    im = plt.imshow(corr_ord.values, vmin=-1, vmax=1, aspect='auto')
    plt.colorbar(im, fraction=0.03)
    plt.xticks(range(len(ordered_cols)), ordered_cols, rotation=90)
    plt.yticks(range(len(ordered_cols)), ordered_cols)
```

```
    plt.title(title)
    plt.tight_layout()
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()
    return ordered_cols

def mutual_info_vs_target(df, target='Fault_Type', n_top=20, outpath=None):
    X = df.drop(columns=[target])
    # impute missing with median
    imp = SimpleImputer(strategy='median')
    X_imp = imp.fit_transform(X)
    y = LabelEncoder().fit_transform(df[target].astype(str))
    mi = mutual_info_classif(X_imp, y, discrete_features=False, random_state=0)
    mi_ser = pd.Series(mi, index=X.columns).sort_values(ascending=False)
    plt.figure(figsize=(8,6))
    mi_ser.head(n_top).plot.bar()
    plt.title('Mutual Information with Fault_Type (top {})'.format(n_top))
    plt.ylabel('MI')
    plt.tight_layout()
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()
    return mi_ser

def physical_vs_fft_corr(df, physical, fft_prefix, outpath=None):
    # collect fft columns that start with fft_prefix
    fft_cols = [c for c in df.columns if c.startswith(fft_prefix)]
    res = {}
    for c in fft_cols:
        # pairwise drop NaNs
        sub = df[[physical, c]].dropna()
        if len(sub) < 3:
            corr = np.nan
        else:
            corr = sub[physical].corr(sub[c])
        res[c] = corr
    res_ser = pd.Series(res).sort_values(key=lambda s: np.abs(s), ascending=False)
    plt.figure(figsize=(8,4))
    res_ser.plot.bar()
    plt.title(f'Correlation between {physical} and {fft_prefix}* bins')
    plt.ylabel('Pearson r')
    plt.xticks(rotation=45)
    plt.tight_layout()
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()
    return res_ser

def pca_loadings(df, n_components=3, outpath=None):
    X = df.drop(columns=['Fault_Type'])
    imp = SimpleImputer(strategy='median')
```

```
    X_imp = imp.fit_transform(X)
    pca = PCA(n_components=n_components)
    Z = pca.fit_transform(X_imp)
    loadings = pd.DataFrame(pca.components_.T, index=X.columns,
                            columns=[f'PC{i+1}' for i in range(n_components)])
    plt.figure(figsize=(10,6))
    sns.heatmap(loadings, annot=True, fmt='.2f', cmap='RdBu_r', center=0)
    plt.title('PCA loadings')
    plt.tight_layout()
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()
    return loadings

def pairplot_top_features(df, features, target='Fault_Type', sample_frac=0.2,
outpath=None):
    # sample if too big
    df_sub = df[[*features, target]].sample(frac=sample_frac, random_state=0) if
sample_frac < 1.0 else df[[*features, target]]
    sns.pairplot(df_sub, hue=target, corner=True, plot_kws={'alpha':0.6, 's':20})
    plt.suptitle('Pairplot for top features', y=1.02)
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()

def rf_feature_importance(df, target='Fault_Type', n_top=20, outpath=None):
    X = df.drop(columns=[target])
    imp = SimpleImputer(strategy='median')
    X_imp = imp.fit_transform(X)
    y = LabelEncoder().fit_transform(df[target].astype(str))
    rf = RandomForestClassifier(n_estimators=200, random_state=0, n_jobs=-1)
    rf.fit(X_imp, y)
    importances = pd.Series(rf.feature_importances_,
index=X.columns).sort_values(ascending=False)
    plt.figure(figsize=(8,6))
    importances.head(n_top).plot.bar()
    plt.title('RandomForest feature importances (top {})'.format(n_top))
    plt.tight_layout()
    if outpath:
        plt.savefig(outpath, dpi=200)
    plt.show()
    # permutation importance (more robust)
    perm = permutation_importance(rf, X_imp, y, n_repeats=10, random_state=0,
n_jobs=-1)
    perm_ser = pd.Series(perm.importances_mean,
index=X.columns).sort_values(ascending=False)
    return importances, perm_ser

if __name__ == '__main__':
    csv_path = '/path/to/Industrial_fault_detection.csv'  # <- put your path here
    df = load_data(csv_path)
    print('Loaded', df.shape)
```

```
# 1. correlation heatmap (clustered)
corr = compute_corr_matrix(df)
ordered = plot_clustered_corr(corr, title='Clustered correlation (all features)')

# 2. mutual info vs Fault_Type
mi_ser = mutual_info_vs_target(df, n_top=30)

# 3. physical vs FFT correlations (examples)
vib_vs_fft = physical_vs_fft_corr(df, 'Vibration', 'FFT_Vib_')
temp_vs_fft = physical_vs_fft_corr(df, 'Temperature', 'FFT_Temp_')
pres_vs_fft = physical_vs_fft_corr(df, 'Pressure', 'FFT_Pres_')

# 4. PCA loadings
loadings = pca_loadings(df, n_components=3)

# 5. pairplot for top 6 features by MI
top_feats = list(mi_ser.index[:6])
pairplot_top_features(df, top_feats, sample_frac=0.3)

# 6. RF importance + permutation
imp_rf, imp_perm = rf_feature_importance(df)
print('Top RF features:', imp_rf.head(10))
print('Top permutation features:', imp_perm.head(10))
```

Пояснение к ключевым блокам кода

- **imputation**: я использую `median` для простоты. Для MI и RF это нормально; для корреляций используется `dropna()` (парные наблюдения) — это даёт более честную оценку r.
- **кластеризация** (`linkage`): расстояние = `1 - |corr|` — это группирует признаки, давая блоки взаимосвязанных признаков.
- **mutual_info_classif** — хороший выбор, он ловит нелинейные связи; выводи топ N признаков и затем визуализируй pairplot.
- **physical_vs_fft_corr** — полезно для определения, какие FFT-бины реально «звучат» в основном сигнале.

Как интерпретировать практические результаты (примеры)

- Если `FFT_Vib_6` имеет высокий |r| с `Vibration` и высокое MI с `Fault_Type=2` (vibration), значит этот FFT-бин — ключевой маркер вибрации: можно агрегировать вокруг этого бина и использовать как отдельный признак.
- Если `Voltage` и `Current` сильно коррелируют (r~0.9), возможно достаточно оставить только `Current` или использовать их соотношение `Power = Voltage * Current`.
- Если группа `FFT_Temp_*` образует плотный кластер и MI с температурными неисправностями высока, то можно посчитать **spectral centroid** или суммарную энергию в диапазоне и использовать её вместо 10 отдельных бинов.

Предупреждения и полезные замечания

- **NaNs**: парные корреляции уменьшают размер выборки; если пропусков много — подумай о продвинутой импутации (KNN, MICE) или об агрегации по кластеру признаков перед imputation.
- **Multicollinearity**: при сильной корреляции между признаками интерпретация коэффициентов линейных моделей становится сложной — используй регуляризацию или RF/PCA.
- **Fault_Type**: если классы несбалансированы, MI и RF могут быть смещены — валидация stratified, взвешивание классов/oversampling (SMOTE) стоит рассмотреть.
- **Временная структура**: все эти методы игнорируют лаги; можно дополнительно посмотреть cross-correlation и лаги между `Flow_Rate` и `Pressure` и т.д.