**Task 1: Find Different Techniques of Data Imputation**

**✅ Project Objective:**

Understand and implement various techniques to handle missing data in datasets to ensure clean input for ML models.

**🛠️ Tools Used:**

* Python
* Pandas
* scikit-learn
* NumPy

**🧠 Techniques:**

1. **Mean/Median/Mode Imputation**

python

CopyEdit

df['Age'].fillna(df['Age'].mean(), inplace=True)

1. **Forward/Backward Fill**

python

CopyEdit

df.fillna(method='ffill')

1. **KNN Imputation**

python

CopyEdit

from sklearn.impute import KNNImputer

imputer = KNNImputer(n\_neighbors=3)

1. **Iterative Imputer (Multivariate Imputation)**

python

CopyEdit

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

1. **Interpolation**

python

CopyEdit

df.interpolate(method='linear')

1. **Deep Learning-based Imputation (Autoencoders)**

**📈 Output:**

Cleaned dataset ready for modeling with minimal information loss.

**🎯 Task 2: What Happens to the Weight of Dropped Category in Categorical Variable**

**✅ Project Objective:**

To understand how one-hot encoding or dummy encoding affects ML models and the role of the dropped category.

**🧠 Explanation:**

* In **one-hot encoding**, one category is dropped to prevent **multicollinearity** (dummy variable trap).
* The **dropped category becomes the baseline**, and all other categories are interpreted **in contrast to it**.
* The weight of the dropped category is implicitly **0**, and other category weights show **difference from it**.

**📝 Task 3: Search About Different Initializers and Their Use Cases & Create a Blog**

**✅ Project Objective:**

Understand weight initializers in deep learning and write a blog explaining their importance and usage.

**🧠 Common Initializers:**

| **Initializer** | **Use Case / Behavior** |
| --- | --- |
| Zeros | Not recommended – causes neurons to learn same features |
| RandomNormal | General use but can lead to exploding/vanishing gradients |
| Xavier (Glorot) | Works well with sigmoid/tanh activations |
| He Initialization | Best for ReLU/LeakyReLU |
| LeCun Initialization | Good for SELU activations |

**📈 Output:**

A technical blog explaining each initializer, with code examples:

python

CopyEdit

from tensorflow.keras.initializers import HeNormal

model.add(Dense(64, activation='relu', kernel\_initializer=HeNormal()))

**🧠 Task 4: Analyze an LLM Model, Its API & Internal Structure. Try to Create Your Own LLM**

**✅ Project Objective:**

Study the architecture and API of an existing Large Language Model (LLM) and attempt to design a basic LLM.

**🛠️ Tools:**

* OpenAI API / Hugging Face Transformers
* TensorFlow / PyTorch

**🧠 Study Example – GPT-2:**

* **Layers**: Transformer decoder layers
* **Activation**: GELU
* **Tokens**: Byte Pair Encoding
* **API**: https://api.openai.com/v1/completions
* **Heads**: Multi-head self-attention

**✅ Create Your Own LLM (Mini version):**

python

CopyEdit

from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from\_pretrained("gpt2")

model = GPT2LMHeadModel.from\_pretrained("gpt2")

Or create from scratch:

python

CopyEdit

model = Sequential([

Embedding(input\_dim=5000, output\_dim=128),

LSTM(256, return\_sequences=True),

Dense(5000, activation='softmax')

])

**📚 Task 5: Find Use Cases of Optimizers & Create a Blog**

**✅ Project Objective:**

Explore optimizers in machine learning, their algorithms, and ideal use cases.

**🧠 Common Optimizers & Use Cases:**

| **Optimizer** | **Use Case / Advantage** |
| --- | --- |
| SGD | Baseline; used in shallow models and large datasets |
| Adam | Popular in deep learning (CNNs, RNNs, Transformers) |
| RMSprop | Good for RNNs and time-series |
| Adagrad | Sparse data like NLP, recommender systems |
| Adadelta | Improves Adagrad with better learning rate decay |

**📝 Example Code:**

python

CopyEdit

model.compile(optimizer='adam', loss='categorical\_crossentropy')

**📈 Output:**

A blog that compares performance and convergence speed across different optimizers.

**🔍 Task 6: Find Which Activation Function Works with Which Type of Pooling**

**✅ Project Objective:**

Understand the compatibility of activation functions and pooling layers in CNNs.

**🧠 Pairing Guidelines:**

| **Activation** | **Pooling** | **Reason** |
| --- | --- | --- |
| ReLU | MaxPooling | Preserves prominent features |
| LeakyReLU / PReLU | MaxPooling | Helps avoid dying ReLU |
| Sigmoid / Tanh | AvgPooling | Smooth activation + smooth pooling |
| Softmax | Usually after Dense, not with pooling |  |

**🔬 Observations:**

* **MaxPooling** + **ReLU** is best for high-dimensional feature extraction (image classification).
* **AvgPooling** + **Sigmoid/Tanh** is better for smoothing features (NLP, certain RNN models).