Summary Report

Core ML (Classification or Regression)
Summary: In this project i build a Basic Sklearn pipeline Artifact model which contain EDA,Feature engineering,Scaling,Imputation below are few details

1.Dataset: Titanic Dataset
2.Features : ['PassengerId', 'Pclass','Name','Sex', 'Age', 'SibSp', 'Parch','Ticket', 'Fare','Cabin', 'Embarked']
2.Target Values: [“Survived”]
3. Exploratory Data Analysis: Univariate,Bivariate,Multivariate
4.Coerelation : Heatmap correlation plot
5.Feature Selection : Dropped lower correlation based on Data format and heatmap reference
6.Splitting : Split train,test with test size of 30% and Random sample seed as 42
7.**Sklearn pipeline** : **Column Transformer**: [Simple Imputer(Imputing Null values/missing values),OneHotencoder(conversion of Categorical data),**Scaling** values to a standard scale using MinMax scaling]>>**Feature Selection** :[Select k best feature]>>**Model**:[Decision Tree]
8. Accuracy: 0.63
Precision:0: 0.64,1:0.58
Recall: 0:0.83,1:34
F1 score: 0:72,1:0.43

2. Edge Detection (Computer Vision)

Edge Detection Evaluation — Classical vs CNN-Based
This script compares classical edge detection algorithms (Sobel, Canny) with a learned CNN-based edge detector trained earlier, using both visual and quantitative evaluations.

Overview
To measure how well a trained CNN edge detector reproduces edge structures compared to classical methods.

Key Steps
Load and preprocess a test image.
Generate edge maps using:
Sobel Operator
Canny Edge Detector
CNN Learned Edge Model
Compare and visualize outputs.
Compute quantitative metrics (MSE, PSNR, SSIM).
Components
1 Classical Edge Detection
Sobel: Uses image gradients in x and y directions to detect edges.
Canny: Multi-stage algorithm involving Gaussian smoothing, gradient computation, and hysteresis thresholding.

Summary Report

2**CNN Edge Detection**
Loads the trained CNN model (SimpleEdgeCNN) from:
"edge_detection/outputs/cnn_edge_model.pth"

Converts test image to tensor and predicts learned edges.
Metrics Used
Metric Description
MSE (Mean Squared Error) Measures pixel-wise difference. Lower = better.
PSNR (Peak Signal-to-Noise Ratio) Indicates reconstruction quality. Higher = better.
SSIM (Structural Similarity Index) Measures structural similarity. Higher = better.

3. Experimentation, Reproducibility & Deployment Pipeline

Experimentation, Reproducibility & Deployment Pipeline/

```
│
├── app/
│   ├── main.py                 # FastAPI app entry point
│
│
├── Data_Ingestion/
│   ├── data_ingestion.py         # Data validation, logging, YAML loading
│   ├── __init__.py
│
├── Model_Training/
│   ├── train.py                # Model training pipeline
│
├── Model_Artifacts_Registry/
│   ├── models/                 # Saved model pipelines
│   └── reports/                # Generated classification reports
│
├── Baseline/train.csv            # Baseline dataset schema reference
├── config/params.yaml            # Model parameters & training configs
├── Monitoring/                 # Logs (validation, errors, training)
│   ├── error.log               # Saved model pipelines
│   └── training.log
│   └── validation.log
├── sapp.py                   # Streamlit frontend app
│
│
├── requirement.txt
└── README.md
```

Summary Report

1.Data ingestion: Injest data,Builds logging folders and log Steps,validates The date with Baseline data
2. Model Train: used same Sklearn pipeline for modeling data but used Param.yaml for split Size and random seed and logged all aspects and stored the model in model artifact after validation of data with column Comparison
Please check the readme file for more info

4:Optimization & Monitoring

Model Optimization & Inference Benchmark — Edge Detection CNN
The script demonstrates how to optimize and benchmark the trained SimpleEdgeCNN model by exporting it to ONNX format and measuring its inference latency on CPU.

5.Quick Reasoning :
Short conceptual Q&A; on CNNs, loss functions, GenAI fine-tuning, drift, and overfitting/underfitting.

Visit the readme inside Quick reasoning for More information