**Final Project Report: Healthcare Predictive Analytics - Stroke Prediction**

**Project Overview**

This project was conducted as part of the Digital Egypt Pioneers Initiative to fulfill the graduation requirements for the AI & Data Science Track. The objective was to develop a predictive model for stroke risk using the "Stroke Prediction Dataset" from Kaggle, aiming to assist healthcare professionals in identifying at-risk patients and improving patient care through data-driven insights. The project was divided into five milestones: Data Collection and Preprocessing, Data Analysis and Visualization, Predictive Model Development, MLOps and Deployment, and Final Documentation.

**Project Milestones**

**Milestone 1: Data Collection, Exploration, and Preprocessing**

* **Objective**: To collect a relevant healthcare dataset, explore its structure, and preprocess it for predictive modeling.
* **Steps and Implementation**
  + **Data Collection**:
    - **Dataset Chosen**: The "Stroke Prediction Dataset" from Kaggle (<https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>).
    - **Details**: The dataset contains 5,110 patient records with 11 features, including age, gender, hypertension, heart\_disease, ever\_married, work\_type, Residence\_type, avg\_glucose\_level, bmi, smoking\_status, and the target variable stroke (0 = No Stroke, 1 = Stroke).
    - **Process**: The dataset file (healthcare-dataset-stroke-data.csv) was downloaded from Kaggle and uploaded to Google Colab for processing.
  + **Data Exploration (EDA)**:
    - **Tools Used**: Python libraries (pandas, matplotlib, seaborn).
    - **Actions**:
      * Loaded the dataset using pandas.read\_csv().
      * Checked the structure with data.info(): 5,110 rows, 12 columns (including id).
      * Generated summary statistics with data.describe(): Average age ~43 years, only 249 stroke cases (4.9%).
      * Identified missing values: bmi had 201 missing entries.
      * Visualized distributions:
        + Histogram of age: Most patients were between 40-60 years.
        + Count plot of stroke: Confirmed class imbalance (4,861 no-stroke vs. 249 stroke).
        + Box plot of age vs. stroke: Stroke patients were older (average ~70 years) compared to non-stroke (~40-50 years).
    - **Observations**:
      * The age distribution is right-skewed, with most patients in their 40s and 50s.
      * The dataset exhibits a significant class imbalance, with stroke cases being relatively rare (4.9%).
      * Stroke patients tend to be older, with an average age around 70, compared to non-stroke patients, who have an average age between 40 and 50.
      * The bmi feature has a right-skewed distribution with some outliers.
      * Categorical features like work\_type and smoking\_status show some imbalance, with 'Private' work type and 'never smoked' being the most frequent categories.
  + **Data Preprocessing**:
    - **Handling Missing Values**:
      * Filled missing bmi values with the mean (~28.9) using data['bmi'].fillna(data['bmi'].mean(), inplace=True).
    - **Encoding Categorical Variables**:
      * Applied Label Encoding to ever\_married (Yes/No → 1/0).
      * Used One-Hot Encoding for gender, work\_type, Residence\_type, and smoking\_status with pd.get\_dummies(drop\_first=True).
    - **Normalization**:
      * Normalized numerical features (age, avg\_glucose\_level, bmi) to a 0-1 range using MinMaxScaler from sklearn.preprocessing.
    - **Dropping Irrelevant Features**:
      * Removed the id column as it was not relevant for modeling.
    - **Output**: Saved the cleaned dataset as cleaned\_stroke\_data.csv.
* **Deliverables**
  + Dataset Exploration Report: Summarized data characteristics, distributions, and issues (e.g., class imbalance, missing values).
  + EDA Notebook: A Jupyter Notebook with visualizations (histograms, count plots, box plots) and summary statistics.
  + Cleaned Dataset: A processed dataset ready for analysis.

**Milestone 2: Data Analysis and Visualization**

* **Objective**: To perform in-depth analysis of the data and create visualizations to highlight trends and insights for healthcare decision-making.
* **Steps and Implementation**
  + **Data Cleaning**:
    - Ensured no remaining missing values or inconsistencies after Milestone 1.
    - Reconfirmed the dataset structure: All features were numerical after encoding.
  + **Data Analysis**:
    - **Correlation Analysis**:
      * Generated a correlation matrix using data.corr() and visualized it with a heatmap (seaborn.heatmap).
      * Findings: age (0.24), hypertension (0.13), and heart\_disease (0.13) showed the strongest positive correlations with stroke.
    - **Feature Importance**:
      * Used a Random Forest model to calculate feature importance.
      * Results: age, avg\_glucose\_level, and bmi were the most influential features for predicting stroke.
  + **Data Visualization**:
    - **Heatmap**: Visualized correlations between features.
    - **Scatter Plot**: Plotted age vs. avg\_glucose\_level with stroke as the hue using seaborn.scatterplot.
      * Insight: Stroke cases were clustered among older patients with higher glucose levels.
    - **Bar Plot**: Showed stroke rates by smoking status (smoking\_status\_smokes).
      * Insight: Smokers had a slightly higher stroke rate.
    - **Interactive Visualization**: Created an interactive scatter plot using plotly.express.scatter for age vs. avg\_glucose\_level, colored by stroke.
* **Deliverables**
  + Cleaned Dataset and Analysis Report: A report detailing the cleaning steps, correlation analysis, and key insights.
  + Visualizations of Health Trends: Heatmaps, scatter plots, bar plots, and an interactive Plotly chart.

**Milestone 3: Predictive Model Development and Optimization**

* **Objective**: To develop and optimize machine learning models for predicting stroke risk.
* **Steps and Implementation**
  + **Model Selection**:
    - Chose two models: Logistic Regression (simple baseline) and Random Forest (ensemble method).
    - Considered the dataset’s characteristics: Class imbalance and numerical features.
  + **Model Training**:
    - Split the data into training (80%) and testing (20%) sets using train\_test\_split(stratify=y) to maintain class distribution.
    - Trained Logistic Regression and Random Forest models using sklearn.
  + **Model Evaluation**:
    - Evaluated initial performance using classification\_report, confusion\_matrix, and roc\_auc\_score.
    - **Initial Results**:
      * Logistic Regression: Accuracy ~95%, but Recall for stroke=1 was 0 (failed to predict stroke cases due to imbalance).
      * Random Forest: Similar issue, with ROC-AUC ~0.80.
  + **Model Optimization**:
    - **Handling Class Imbalance**:
      * Applied SMOTE (Synthetic Minority Oversampling Technique) using imblearn.over\_sampling.SMOTE to balance the training data.
      * Post-SMOTE: Training data had equal numbers of stroke=0 and stroke=1.
    - **Retraining**:
      * Retrained Random Forest on the balanced data.
      * New Results: Recall for stroke=1 improved to ~0.4, ROC-AUC increased to ~0.85.
    - **Hyperparameter Tuning**:
      * Used GridSearchCV to tune Random Forest parameters (n\_estimators, max\_depth).
      * Best parameters: n\_estimators=200, max\_depth=20.
  + **Metrics**:
    - Focused on Recall and ROC-AUC due to class imbalance. High Recall is crucial to minimize false negatives (i.e., failing to identify stroke patients). ROC-AUC provides a measure of the model's ability to discriminate between the two classes.
    - Other metrics such as precision and F1-score were also considered.
* **Deliverables**
  + Predictive Model Performance Report: Detailed comparison of model performance before and after optimization.
  + Model Code: Python code for training, evaluation, and optimization.
  + Final Model: Optimized Random Forest model saved as stroke\_rf\_model.pkl.

**Milestone 4: MLOps, Deployment, and Monitoring**

* **Objective**: To implement MLOps practices, deploy the model for real-world use, and set up monitoring.
* **Steps and Implementation**
  + **MLOps Implementation**:
    - **Tool Used**: MLflow for experiment tracking.
    - **Process**:
      * Installed MLflow in Colab (!pip install mlflow).
      * Set up an experiment (mlflow.set\_experiment("Stroke\_Prediction\_Experiment")).
      * Logged parameters (model\_type, random\_state), metrics (accuracy, f1\_score), and the model with mlflow.sklearn.log\_model.
      * Added an input\_example to avoid MLflow warnings about model signatures.
    - **Challenge**: MLflow UI couldn’t be fully utilized in Colab due to server limitations.
  + **Model Deployment**:
    - **Approach**: Created a Flask API to serve predictions.
    - **Code**:
    - from flask import Flask, request, jsonify
    - import joblib
    - import pandas as pd
    - app = Flask(\_\_name\_\_)
    - model = joblib.load("stroke\_rf\_model.pkl")
    - @app.route('/predict', methods=['POST'])
    - def predict():
    - data = request.get\_json(force=True)
    - df = pd.DataFrame([data])
    - prediction = model.predict(df)
    - return jsonify({'prediction': int(prediction[0])})
    - if \_\_name\_\_ == '\_\_main\_\_':
    - app.run(debug=True, host='0.0.0.0', port=5000)
    - **Testing**:
      * Ran the Flask app locally (python app.py).
      * Sent POST requests using curl or Python requests with sample patient data.
      * Example response: {"prediction": 1}.
    - **API Details**:
      * The API is designed to receive patient data in JSON format via a POST request to the /predict endpoint.
      * The expected JSON input should contain the following keys, matching the feature names in the dataset: age, gender, hypertension, heart\_disease, ever\_married, work\_type, Residence\_type, avg\_glucose\_level, bmi, and smoking\_status.
      * The API returns a JSON response with a single key, prediction, which is an integer: 0 for "No Stroke" and 1 for "Stroke".
    - **Integration into Healthcare Systems**:
      * This API can be integrated into Electronic Health Record (EHR) systems. A user interface within the EHR would allow doctors to input patient information, and the system would send this data to the API. The API's response (the stroke risk prediction) would then be displayed to the doctor within the EHR interface, aiding in clinical decision-making.
    - **Security Considerations**:
      * In a real-world healthcare setting, security is paramount. The API should be protected with authentication mechanisms (e.g., API keys, OAuth 2.0) to ensure that only authorized systems can access it. Data transmission should occur over HTTPS to encrypt sensitive patient information. The API should also comply with relevant data privacy regulations (e.g., HIPAA).
    - **Error Handling**:
      * The API should include robust error handling to manage invalid input data or unexpected issues. Meaningful error messages should be returned to the client to facilitate debugging.
    - **Scalability**:
      * For high-traffic healthcare systems, the API may need to be deployed on a scalable infrastructure (e.g., cloud-based servers) to handle a large number of requests concurrently.
    - **Challenge**: Encountered a TemplateNotFound: index.html error when accessing the root URL (/). Resolved by confirming the API-only design and using POST requests to /predict.
  + **Model Monitoring**:
    - **Approach**: Simulated monitoring by evaluating the model on new data.
    - **Code**:
      * Used test data as a proxy for new data.
      * Calculated accuracy and set a threshold (e.g., 80%) to trigger retraining alerts.
    - **Real-world monitoring**:
      * In a production environment, a dedicated monitoring system would track the model's performance on live data. This system would collect data on actual patient outcomes and compare them to the model's predictions.
      * Tools like Prometheus and Grafana can be used to visualize key metrics (e.g., accuracy, precision, recall) over time. Alerts can be configured to notify the data science team if performance drops below a predefined threshold, indicating potential model drift.
    - **Data Drift**:
      * Model performance can degrade over time due to changes in the underlying data distribution, a phenomenon known as data drift. For example, changes in population demographics, diagnostic practices, or other factors could affect the relationship between the input features and the target variable (stroke).
      * To detect data drift, statistical methods can be employed to compare the distribution of new data to the distribution of the training data. If significant drift is detected, the model should be retrained with the new data.
    - **Retraining**:
      * The frequency of retraining depends on the rate of data drift and the criticality of the model's predictions. In a healthcare setting, where accuracy is crucial, the model should be retrained regularly (e.g., every 3-6 months) or whenever significant data drift is detected. The retraining process should involve collecting new labeled data, updating the model's parameters, and validating its performance on a hold-out set.
* **Deliverables**
  + Deployed Predictive Model: Flask API for stroke prediction.
  + MLOps Report: Summary of MLflow usage and experiment tracking.
  + Model Monitoring Setup: Code for performance tracking and retraining alerts.

**Milestone 5: Final Documentation and Presentation**

* **Objective**: To document the project comprehensively and present the results to healthcare stakeholders.
* **Steps and Implementation**
  + **Final Report**:
    - Compiled a detailed report (this document) summarizing all milestones, challenges, and insights.
    - Included recommendations for healthcare integration and future improvements.
  + **Final Presentation**:
    - **Tools Used**: Plotly for interactive visualizations, prepared for PowerPoint.
    - **Content**:
      * Introduction: Overview of stroke prediction.
      * Data Insights: Visualizations (e.g., scatter plot of age vs. avg\_glucose\_level).
      * Model Performance: Highlighted ROC-AUC of 0.85 with Random Forest.
      * Deployment: Demonstrated the Flask API.
      * Future Steps: Suggested real-time integration and monitoring.
    - **Code**:
    - import plotly.express as px
    - fig = px.scatter(data, x='age', y='avg\_glucose\_level', color='stroke',
    - title="Age vs. Glucose Level by Stroke Risk")
    - fig.show()
* **Deliverables**
  + Final Project Report: This comprehensive document.
  + Final Presentation: Interactive visualizations and a presentation outline for stakeholders.

**Challenges Faced**

* **Class Imbalance**: The dataset had only 249 stroke cases out of 5,110, leading to poor model performance on the minority class. Addressed using SMOTE.
* **Deployment in Colab**: Flask deployment required a local setup due to Colab’s server limitations. Resolved by running the API locally.
* **MLflow Warning**: Encountered a warning about missing model signatures. Fixed by adding an input\_example in MLflow logging.
* **Data Quality**: The dataset contained missing values for the bmi feature, which required imputation. The accuracy of the model depends on the quality and completeness of the input data.

**Key Insights**

* Age is the most significant predictor of stroke risk, followed by avg\_glucose\_level and hypertension.
* Balancing the dataset with SMOTE significantly improved model performance, though it increased false positives.
* The model can be integrated into healthcare systems to provide real-time risk assessments.

**Recommendations**

* **Integration**: Deploy the model in hospitals via a user-friendly interface for doctors to input patient data and receive predictions.
* **Monitoring**: Set up a live monitoring system to detect model drift and retrain with new data periodically.
* **Future Work**:
  + Explore additional features (e.g., genetic data, detailed patient history) and advanced models (e.g., Neural Networks) to improve accuracy.
  + Investigate the impact of different data imputation methods for handling missing bmi values.
  + Conduct a thorough analysis of potential biases in the dataset and implement techniques to mitigate them.
  + Explore the use of advanced feature engineering techniques to create more informative features.
  + Implement a system to retrain the model every 3-6 months, or whenever significant data drift is detected.

**Conclusion**

The Healthcare Predictive Analytics project successfully developed a stroke prediction model with a ROC-AUC of ~0.85, offering valuable insights for healthcare professionals. By following a structured approach across five milestones, the project addressed data challenges, built an optimized model, and deployed it for practical use, paving the way for improved patient care.