Breast Cancer detection and prediction

1. Data Ingestion into Snowflake

1. **Upload the data from Kaggle to Snowflake**:
    - Download the CSV file ("Breast cancer detection.csv") from Kaggle.
    - Use the Snowflake command line interface (SnowSQL) to upload the file to a Snowflake stage:

sql

Copy code

```
-- Create a new table to hold the data
CREATE OR REPLACE TABLE breast_cancer_data (
  id INT,
  diagnosis VARCHAR,
  radius_mean FLOAT,
  texture_mean FLOAT,
  perimeter_mean FLOAT,
  area_mean FLOAT,
  smoothness_mean FLOAT,
  compactness_mean FLOAT,
  concavity_mean FLOAT,
  concave_points_mean FLOAT,
  -- Continue listing all the columns based on the CSV file
  fractal_dimension_worst FLOAT
);


-- Load the data into Snowflake
PUT file://<local_file_path>/Breast_cancer_detection.csv @mystage;
```

```
COPY INTO breast_cancer_data
FROM @mystage/Breast_cancer_detection.csv
FILE_FORMAT = (TYPE = 'CSV' FIELD_OPTIONALLY_ENCLOSED_BY = '"');
```

2. **Check the data in Snowflake**:
    - Run SQL queries to ensure the data is properly loaded.

```
sql
Copy code
SELECT * FROM breast_cancer_data LIMIT 10;
```

2. Exploratory Data Analysis (EDA)

1. **Check the data**:
    - View the first few rows of data in Snowflake to understand the structure and types of variables:

```
sql
Copy code
SELECT * FROM breast_cancer_data LIMIT 5;
```

2. **Descriptive statistics**:
    - Calculate basic statistics like mean, median, and standard deviation to get a sense of the data distribution:

```
sql
Copy code
SELECT
 AVG(radius_mean),
 AVG(texture_mean),
 AVG(perimeter_mean),
 STDDEV(area_mean)
FROM breast_cancer_data;
```

3. **Check for missing data**:
    - You will want to check for any null values and handle them if necessary:

sql

Copy code

SELECT COUNT(*) FROM breast_cancer_data WHERE radius_mean IS NULL;

4. **Distribution of the Target Variable (diagnosis)**:
    - M stands for Malignant and B stands for Benign.
    - Count the number of cases for each diagnosis type:

sql

Copy code

SELECT diagnosis, COUNT(*) FROM breast_cancer_data GROUP BY diagnosis;

---

3. Data Preprocessing

1. **Label Encoding**:
    - Convert the categorical variable diagnosis into numeric values (M = 1, B = 0) for modeling purposes:

sql

Copy code

-- In Snowflake SQL, you would handle this after exporting the data to a processing environment like Python

-- OR, you can use a CASE WHEN clause to create a new column

ALTER TABLE breast_cancer_data ADD labels INT;

UPDATE breast_cancer_data

SET labels = CASE WHEN diagnosis = 'M' THEN 1 ELSE 0 END;

2. **Standardize the Data**:

- Use SQL to normalize/standardize features using mean and standard deviation, or use Python for better control:

sql

Copy code

```
-- Calculate z-scores for each feature
SELECT
  (radius_mean - AVG(radius_mean) OVER()) / STDDEV(radius_mean) OVER() AS radius_mean_zscore,
  (texture_mean - AVG(texture_mean) OVER()) / STDDEV(texture_mean) OVER() AS texture_mean_zscore
FROM breast_cancer_data;
```

---

4. Model Building and Training

1. **Train/Test Split**:
    - Split the data into training and testing sets using a 70/30 split. This step is done in Python, outside of Snowflake:

python

Copy code

```
from sklearn.model_selection import train_test_split
x = data.drop(columns='labels')
y = data['labels']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
```

2. **Neural Network with TensorFlow**:
    - The following neural network was built using TensorFlow to predict breast cancer.

python

Copy code

```
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(30,)),
  keras.layers.Dense(20, activation='relu'),
  keras.layers.Dense(2, activation='sigmoid')
])


model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])


history = model.fit(x_train_std, y_train, validation_split=0.1, epochs=10)
```

3. **Evaluate Model Performance**:
    - Evaluate the model's performance using the testing dataset:

python

Copy code

```
loss, accuracy = model.evaluate(x_test_std, y_test)
print(f"Model accuracy: {accuracy:.4f}")
```

    - The reported accuracy on the test data was **95.30%**.

---

5. Prediction

The model predicts the likelihood of each class (Benign or Malignant) using a prediction probability for each class. The class with the higher probability is selected as the predicted label.

python

Copy code

```
input_data_std = scaler.transform(input_data_reshaped)
prediction = model.predict(input_data_std)
```

prediction_label = [np.argmax(prediction)]

For the sample input, the model predicted **Malignant** tumor with a confidence score of 0.969 for class 0 (Malignant).

Numerical Results and Conclusion

- **Model Accuracy**: The model achieved an accuracy of **95.30%** on the test dataset.
- **Key Insights**:
    - The model is highly effective at predicting breast cancer based on the given dataset.
    - Malignant cases generally have higher means for most features compared to benign cases, which was observed during the exploratory data analysis.