🧾 Post-Session Notes: Revision – Evaluating Metrics and Result Analysis, Testing and Training

---

🩺 1. Project Introduction & Evaluation Context

- **The project focuses on analyzing healthcare-related data to uncover patterns and model relationships in variables like:**
  - **Hours of sleep**
  - **Sleep quality**
  - **Stress level**
  - **Daily steps**
  - **Calorie intake**
- **This session emphasized the importance of interpreting model results, handling class imbalance, and evaluating performance with appropriate metrics.**
- **Earlier steps like data preprocessing, visualization, and basic modeling (SVM, Logistic Regression) laid the groundwork.**

🧠 2. Clustering Techniques: K-Means & DBSCAN

🔹 **K-Means Clustering**

- Goal: Group data into clusters based on distance to centroids (means).
- Key parameters:
  - `n_clusters`: Predefined number of clusters.
  - `random_state`: For reproducibility.
- Application:
  - Used on features like sleep hours, sleep quality, and stress level.
  - Helped in identifying behavioral patterns among users.
- Evaluation with Silhouette Score:
  - Range: -1 to 1
  - >0.5 = good clustering
  - ~0.3 = weak but acceptable
  - Clusters showed overlap and scattering, indicating complex relationships not fully captured by K-means.

🔹 DBSCAN Clustering

- Goal: Density-based clustering that doesn't require predefined clusters.
- Key Parameters:
  - `eps`: Maximum distance to consider for neighborhood.

- `min_samples`: Minimum points to form a cluster.

- **Advantages:**
  - Handles noise/outliers explicitly.
  - Finds non-linear, irregularly shaped clusters.

- **Application:**
  - Applied on calorie intake and daily steps.
  - Resulted in clusters + noise points.
  - Tuning `eps` drastically affected number and quality of clusters.

📏 3. Evaluation Metrics: Going Beyond Accuracy

🔸 **Why Accuracy Isn't Enough**

- In imbalanced datasets, high accuracy can be misleading.
- E.g., 98% accuracy may hide the fact that the minority class is barely predicted.

🔸 **Key Metrics:**

- **Precision:** How many predicted positives are actually correct?
  (Focus: False Positives)

- **Recall**: How many actual positives are correctly predicted?
  (Focus: False Negatives)
- **F1 Score**: Harmonic mean of precision and recall – balances both.
- **Confusion Matrix**: Breaks down predictions into:
  - True Positives
  - False Positives
  - True Negatives
  - False Negatives

Used a logistic regression model to demonstrate how a model could score high accuracy but low precision/recall for minority classes.

⚖️ 4. Handling Class Imbalance

- Imbalanced data (e.g., very few samples in one class) affects model performance and metric fairness.
- **Solutions**:
  - Use `class_weight='balanced'` in models (like Logistic Regression) to penalize majority class overconfidence.
  - Consider oversampling or undersampling (not covered deeply here but conceptually important).

```
model = LogisticRegression(class_weight='balanced')
model.fit(X_train, y_train)
```

🔁 5. Train-Test Split vs Cross-Validation

🔸 Train-Test Split

- Simple partition of data into training and testing subsets.
- Can lead to biased results depending on the split.

🔸 Cross-Validation

- K-Fold CV splits data into multiple subsets.
- Each fold is used once as a test set, others as training.
- Reduces overfitting risk and provides average metric performance.

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X, y, cv=5, scoring='f1')
print("Mean F1 score:", scores.mean())
```

- Epoch vs Cross-Validation:
  - Epoch: One full pass through training data (used in deep learning).
  - Cross-validation: Evaluation technique to test model robustness.

🧩 6. Interpreting Clustering Output

- Cluster results from both KMeans and DBSCAN require domain interpretation:
  - What does each cluster represent?
  - How does it help in understanding user behavior?
- Challenges:
  - Overlapping clusters
  - Arbitrary shapes in data
  - Defining meaningful cluster labels

📌 7. Summary of Key Coding Techniques

```
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score, confusion_matrix, f1_score
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression

# KMeans Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
silhouette = silhouette_score(X, labels)

# DBSCAN Clustering
dbscan = DBSCAN(eps=0.5, min_samples=20)
```

```
dbscan_clusters = dbscan.fit_predict(X)

# Logistic Regression with Class Weights
logreg = LogisticRegression(class_weight='balanced')
logreg.fit(X_train, y_train)

# Cross-validation
f1_scores = cross_val_score(logreg, X, y, cv=5, scoring='f1')
```

🧠 8. Key Takeaways & Best Practices

- Never rely solely on accuracy, especially with imbalanced data.
- Always look at precision, recall, and F1 for classification models.
- Silhouette score is an effective way to evaluate clustering quality.
- Experiment with clustering parameters and features.
- DBSCAN is powerful for noisy, irregular data.
- Cross-validation offers reliable performance assessment.
- Model evaluation is not just technical — it needs contextual understanding.

🎯 Conclusion

This session integrated clustering techniques, evaluation metrics, and testing strategies into the broader healthcare dataset project. It emphasized

robustness, interpretability, and thoughtful metric selection—all vital in building real-world, dependable ML models.
