Mock Test Report

Student: Jyotiraditya Saha

Date: March 30, 2025

10
out of 50

Suggestions to Improve

The student demonstrates a foundational understanding of machine learning concepts but needs to strengthen their knowledge in data preprocessing, model evaluation, hyperparameter tuning, and the interpretation of model behavior in specific contexts.

Topic: Data Preprocessing and Scaling - Review the different types of feature scaling techniques (StandardScaler, MinMaxScaler, MaxAbsScaler) and when to apply each. Practice applying these techniques to datasets with varying characteristics.

Topic: Recurrent Neural Networks (RNNs) - Study the `return_sequences` parameter in Keras RNN layers in detail. Understand its effect on the output shape and how it influences the architecture of stacked RNNs (sequence-to-sequence vs. sequence-to-vector).

Topic: Hyperparameter Tuning and Model Selection - Deepen your understanding of `GridSearchCV` in scikit-learn. Practice using it with different models and parameter grids. Focus on interpreting the results (best parameters, best score, etc.).

Topic: Linear Regression and Multicollinearity - Learn about the effects of multicollinearity in linear regression models. Understand how it impacts model interpretability and stability. Explore techniques for addressing multicollinearity (e.g., feature selection, regularization).

Topic: One-Hot Encoding - Review the purpose of one-hot encoding for categorical features. Practice using `pd.get_dummies` in pandas to create one-hot encoded representations of categorical variables. Understand how this impacts model training.

Topic: Model Evaluation Metrics for Imbalanced Datasets - Study the strengths and weaknesses of different evaluation metrics (accuracy, precision, recall, F1-score) in the context of imbalanced datasets. Understand why accuracy can be misleading in such cases and the importance of precision and recall.

Topic: TensorFlow/Keras Model Usage - Review the use of `np.newaxis` in NumPy to add a dimension to an array, and understand why this is necessary when feeding single observations into a TensorFlow/Keras model.

Topic: Learning Rate Schedules - Study different learning rate schedules, including exponential decay. Understand how they affect the training process and model convergence. Practice

implementing different learning rate schedules in your training loops.

Practice coding the examples in the questions and experiment with variations.

Use online resources (documentation, tutorials, forums) to clarify any remaining doubts.

Work on a variety of datasets to gain practical experience.

Correct Answers

Question	Correct Answer
What will be the output of the following code? ```python import numpy as np from sklearn.preprocessing import StandardScaler data = np.array([[1, 2], [3, 4], [5, 6]]) scaler = StandardScaler() scaled_data = scaler.fit_transform(data) print(scaled_data) ```	A NumPy array with zero mean and unit variance
Which of the following best describes the purpose of the `return_sequences` parameter in a Keras RNN layer? ```python from tensorflow.keras.layers import LSTM lstm_layer = LSTM(units=64, return_sequences=True, input_shape=(timesteps, features)) ```	It specifies whether the RNN layer should return the full sequence of hidden states or only the last hidden state.
Consider the following code snippet. What is the primary purpose of the `GridSearchCV` object? ```python from sklearn.model_selection import GridSearchCV from sklearn.ensemble import RandomForestClassifier param_grid = {'n_estimators': [100, 200], 'max_depth': [None, 10]} rf = RandomForestClassifier() grid_search = GridSearchCV(rf, param_grid, cv=5) ```	To automatically select the best hyperparameters for a given model using cross-validation.
The following code trains a simple linear regression model. What will likely happen if the training data contains significant multicollinearity? ```python from sklearn.linear_model import LinearRegression import numpy as np X = np.array([[1, 2], [3, 4], [5, 6]]) # features with high correlation y = np.array([7, 9, 11]) model = LinearRegression() model.fit(X, y) ```	The model's coefficients will be highly unstable and difficult to interpret.
What does the following code snippet achieve? ```python import pandas as pd data = {'col1': [1, 2, 3], 'col2': ['A', 'B', 'A']} df = pd.DataFrame(data) dummies = pd.get_dummies(df['col2']) ```	It applies one-hot encoding to the 'col2' column.
Which metric is MOST appropriate to evaluate a binary classification model where the classes are heavily imbalanced (e.g., 99% negative, 1% positive)? ```python from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score ```	Recall
Given the following code, what does `model(obs[np.newaxis])` accomplish? ```python import numpy as np import tensorflow as tf class	

MyModel(tf.keras.Model): def __init__(self): super(MyModel, self).__init__() self.dense = tf.keras.layers.Dense(1, activation='sigmoid') def call(self, x): return self.dense(x) model = MyModel() obs = np.array([0.5]) output = model(obs[np.newaxis]) ```	It reshapes the observation to fit the model's input shape requirement.
What is the primary purpose of the `tf.GradientTape()` context manager in TensorFlow? ```python import tensorflow as tf with tf.GradientTape() as tape: # ... some computations ... gradients = tape.gradient(loss, variables) ```	To record operations for automatic differentiation and compute gradients.
The below code uses `exponential_decay_fn`. What learning rate schedule does this represent? ```python def exponential_decay_fn(epoch, lr): return lr * 0.1**(1 / 20) ```	An exponential decay schedule
What is the primary role of the `discount_factor` in the following `discount_rewards` function? ```python import numpy as np def discount_rewards(rewards, discount_factor): discounted = np.array(rewards) for step in range(len(rewards) - 2, -1, -1): discounted[step] += discounted[step + 1] * discount_factor return discounted ```	It reduces the influence of future rewards on the current action's evaluation.