Machine Learning Algorithms - Practical Test Paper

Test Information

Total Marks: 100
Duration: 3 Hours
Test Type: Practical Coding Assessment

Instructions

1. All questions require you to write complete Python code
2. Use the specified Kaggle datasets for each question
3. Include data loading, preprocessing, model training, and evaluation
4. Print appropriate evaluation metrics as specified
5. Save your code as separate `.py` or `.ipynb` files for each question
6. Comment your code appropriately
7. Handle errors and edge cases

Required Libraries

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import *
```

Section A: Linear Regression (25 Marks)

Question 1: House Price Prediction (15 Marks)

Dataset: House Prices - Advanced Regression Techniques
File: `train.csv`

Task: Build a Linear Regression model to predict house prices (`SalePrice`).

Requirements:

1. Load the dataset from Kaggle
2. Select these numerical features: `OverallQual`, `GrLivArea`, `GarageCars`, `GarageArea`, `TotalBsmtSF`, `1stFlrSF`, `FullBath`, `TotRmsAbvGrd`, `YearBuilt`
3. Handle any missing values in these features (use mean imputation)
4. Split data into 80% train and 20% test sets (random_state=42)
5. Train a Linear Regression model
6. Calculate and print:
    - R^2 Score (training and testing)
    - RMSE (Root Mean Squared Error)
    - MAE (Mean Absolute Error)
7. Create a scatter plot of Actual vs Predicted prices for test set
8. Print the coefficients of all features

Expected Output:

```
Model Performance:
 Training R² Score: X.XXXX
 Testing R² Score: X.XXXX
 RMSE: $XX,XXX.XX
 MAE: $XX,XXX.XX

Feature Coefficients:
 OverallQual: XXXXX.XX
 GrLivArea: XXX.XX
 ...
```

Grading:

- Data loading and preprocessing (3 marks)
- Model training (4 marks)
- Evaluation metrics (4 marks)
- Visualization (2 marks)
- Code quality and comments (2 marks)

Question 2: Medical Insurance Cost Prediction (10 Marks)

Dataset: Medical Cost Personal Datasets
File: `insurance.csv`

Task: Predict insurance charges based on personal information.

Requirements:

1. Load the dataset
2. Convert categorical features (`sex`, `smoker`, `region`) to numerical using Label Encoding
3. Use all features except `charges` as inputs
4. Target variable is `charges`
5. Split data 70-30 (random_state=42)
6. Train Linear Regression model
7. Print:
    - R^2 Score
    - RMSE
    - Prediction for a new person: age=30, sex='male', bmi=25.5, children=2, smoker='yes', region='northwest'

Expected Output:

```
Model Performance:
R² Score: X.XXXX
RMSE: $X,XXX.XX

New Prediction:
Predicted Insurance Cost: $XX,XXX.XX
```

Grading:

- Data preprocessing with encoding (3 marks)
- Model training (3 marks)
- Evaluation and new prediction (3 marks)
- Code quality (1 mark)

Section B: Logistic Regression (25 Marks)

Question 3: Titanic Survival Prediction (15 Marks)

Dataset: Titanic - Machine Learning from Disaster
File: `train.csv`

Task: Build a Logistic Regression classifier to predict survival.

Requirements:

1. Load the Titanic dataset
2. Select features: `Pclass`, `Sex`, `Age`, `SibSp`, `Parch`, `Fare`, `Embarked`
3. Handle missing values:
   - `Age`: Fill with median
   - `Embarked`: Fill with most frequent value
4. Convert categorical features:
   - `Sex`: male=1, female=0
   - `Embarked`: Use Label Encoding
5. Target variable: `Survived`
6. Split data 75-25 (random_state=42)
7. Scale features using StandardScaler
8. Train Logistic Regression with max_iter=1000
9. Print:
   - Accuracy, Precision, Recall, F1-Score
   - Confusion Matrix
   - Classification Report
10. Create confusion matrix heatmap

Expected Output:

```
Model Performance:
  Accuracy: X.XXXX
  Precision: X.XXXX
  Recall: X.XXXX
  F1-Score: X.XXXX

Confusion Matrix:
  [[TN FP]
   [FN TP]]

Classification Report:
  ...
```

Grading:

- Data preprocessing (4 marks)
- Feature scaling (2 marks)
- Model training (3 marks)
- Evaluation metrics (4 marks)
- Visualization (2 marks)

Question 4: Heart Disease Prediction (10 Marks)

Dataset: Heart Disease UCI
File: `heart.csv`

Task: Predict presence of heart disease.

Requirements:

1. Load dataset
2. Use all features except `target` as inputs
3. Target variable: `target` (0 = no disease, 1 = disease)
4. Split data 80-20 (random_state=42)
5. Scale features
6. Train Logistic Regression
7. Calculate and print:
    - Accuracy

- ROC-AUC Score
- Confusion Matrix

8. Plot ROC Curve
9. Predict for a new patient with these values: age=50, sex=1, cp=2, trestbps=130, chol=250, fbs=0, restecg=1, thalach=150, exang=0, oldpeak=1.5, slope=2, ca=0, thal=2

Expected Output:

```
Model Performance:
 Accuracy: X.XXXX
 ROC-AUC Score: X.XXXX

Confusion Matrix:
 [[TN FP]
  [FN TP]]

New Patient Prediction:
 Heart Disease Risk: X.XX%
 Classification: Disease/No Disease
```

Grading:

- Data preprocessing and scaling (2 marks)
- Model training (2 marks)
- Evaluation metrics (3 marks)
- ROC curve visualization (2 marks)
- New prediction (1 mark)

Section C: Decision Trees & Random Forest (25 Marks)

Question 5: Iris Species Classification (12 Marks)

Dataset: Iris Species
File: Iris.csv

Task: Build both Decision Tree and Random Forest classifiers.

Requirements:

1. Load the Iris dataset

2. Use features: `SepalLengthCm`, `SepalWidthCm`, `PetalLengthCm`, `PetalWidthCm`
3. Target: `Species`
4. Split data 70-30 (random_state=42)
5. Train Decision Tree (max_depth=5, random_state=42)
6. Train Random Forest (n_estimators=100, max_depth=5, random_state=42)
7. Print for BOTH models:
    - Accuracy
    - Classification Report
    - Feature Importance
8. Compare performance
9. Create confusion matrix for both models

Expected Output:

```
Decision Tree Performance:
  Accuracy: X.XXXX

Feature Importances:
  PetalLengthCm: X.XXXX
  PetalWidthCm: X.XXXX
  ...

Random Forest Performance:
  Accuracy: X.XXXX

Feature Importances:
  PetalLengthCm: X.XXXX
  ...

Performance Improvement: XX.XX%
```

Grading:

- Data loading and preprocessing (2 marks)
- Decision Tree implementation (3 marks)
- Random Forest implementation (3 marks)
- Comparison and analysis (2 marks)
- Feature importance display (2 marks)

Question 6: Loan Approval Prediction (13 Marks)

Dataset: Loan Prediction Dataset
File: `train_loan.csv`

Task: Predict loan approval status using Random Forest.

Requirements:

1. Load the dataset
2. Select features: `Gender`, `Married`, `Dependents`, `Education`, `Self_Employed`, `ApplicantIncome`, `CoapplicantIncome`, `LoanAmount`, `Loan_Amount_Term`, `Credit_History`, `Property_Area`
3. Handle missing values appropriately
4. Encode categorical variables using Label Encoding
5. Target: `Loan_Status` (Y/N)
6. Split data 80-20 (random_state=42)
7. Train Random Forest with:
    - n_estimators=200
    - max_depth=10
    - min_samples_split=10
    - random_state=42
8. Use GridSearchCV to find optimal n_estimators (try: 100, 200, 300) and max_depth (try: 5, 10, 15)
9. Print:
    - Best parameters
    - Best cross-validation score
    - Test accuracy
    - Feature importance (top 5 features)
    - Confusion matrix

Expected Output:

```
Grid Search Results:
  Best n_estimators: XXX
  Best max_depth: XX
  Best CV Score: X.XXXX

Final Model Performance:
  Test Accuracy: X.XXXX

Top 5 Important Features:
  1. Credit_History: X.XXXX
  2. ApplicantIncome: X.XXXX
  ...

Confusion Matrix:
  [[TN FP]
   [FN TP]]
```

Grading:

- Data preprocessing (3 marks)
- Random Forest implementation (3 marks)
- GridSearchCV implementation (3 marks)
- Evaluation metrics (2 marks)
- Feature importance (2 marks)

Section D: K-Nearest Neighbors (25 Marks)

Question 7: Wine Quality Classification (12 Marks)

Dataset: Red Wine Quality
File: `winequality-red.csv`

Task: Classify wine quality using KNN.

Requirements:

1. Load the dataset
2. Create binary classification: quality >= 6 as "Good" (1), else "Bad" (0)
3. Use all features except `quality` as inputs
4. Split data 75-25 (random_state=42)

5. **Scale features using StandardScaler** (CRITICAL!)
6. Test K values from 1 to 30
7. Plot accuracy vs K value
8. Find optimal K using cross-validation
9. Train KNN with optimal K
10. Print:
    - Optimal K value
    - Accuracy, Precision, Recall, F1-Score
    - Confusion Matrix
11. Compare `weights='uniform'` vs `weights='distance'`

Expected Output:

```
K Value Analysis:
  Optimal K: XX
  Best CV Accuracy: X.XXXX

Model Performance (Uniform Weights):
  Accuracy: X.XXXX
  Precision: X.XXXX
  Recall: X.XXXX
  F1-Score: X.XXXX

Model Performance (Distance Weights):
  Accuracy: X.XXXX
  Improvement: X.XX%

Confusion Matrix:
  [[TN FP]
   [FN TP]]
```

Grading:

- Data preprocessing (2 marks)
- Feature scaling (2 marks)
- K value optimization (3 marks)
- Model training (2 marks)
- Evaluation and comparison (2 marks)

- Visualization (1 mark)

Question 8: Diabetes Prediction (13 Marks)

Dataset: Pima Indians Diabetes Database
File: `diabetes.csv`

Task: Predict diabetes using KNN with hyperparameter tuning.

Requirements:

1. Load the dataset
2. Features: `Pregnancies`, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, `DiabetesPedigreeFunction`, `Age`
3. Target: `Outcome` (0 = no diabetes, 1 = diabetes)
4. Handle zero values in `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI` by replacing with median
5. Split data 70-30 (random_state=42)
6. Scale features using StandardScaler
7. Use GridSearchCV to find optimal:
    - n_neighbors: [3, 5, 7, 9, 11, 13, 15]
    - weights: ['uniform', 'distance']
    - metric: ['euclidean', 'manhattan']
8. Use 5-fold cross-validation
9. Print:
    - Best hyperparameters
    - Best CV score
    - Test accuracy
    - Classification report
    - Confusion matrix heatmap
10. Predict for a new patient: Pregnancies=2, Glucose=120, BloodPressure=70, SkinThickness=25, Insulin=100, BMI=28.5, DiabetesPedigreeFunction=0.5, Age=35

Expected Output:

```
Grid Search Results:
  Best K: XX
  Best Weights: XXXXXXX
  Best Metric: XXXXXXX
  Best CV Score: X.XXXX

Final Model Performance:
  Test Accuracy: X.XXXX
  Precision: X.XXXX
  Recall: X.XXXX
  F1-Score: X.XXXX

Classification Report:
  ...

Confusion Matrix:
  [[TN FP]
   [FN TP]]

New Patient Prediction:
  Diabetes Risk: XX.XX%
  Classification: Diabetes/No Diabetes
  Confidence: XX.XX%
```

Grading:

- Data preprocessing (3 marks)
- Handling missing values (2 marks)
- Feature scaling (1 mark)
- GridSearchCV implementation (3 marks)
- Model evaluation (2 marks)
- Visualization (1 mark)
- New prediction (1 mark)

Submission Guidelines

File Naming Convention

```
Question1_LinearRegression_HousePrices.ipynb
Question2_LinearRegression_Insurance.ipynb
Question3_LogisticRegression_Titanic.ipynb
Question4_LogisticRegression_HeartDisease.ipynb
Question5_DecisionTree_Iris.ipynb
Question6_RandomForest_LoanApproval.ipynb
Question7_KNN_WineQuality.ipynb
Question8_KNN_Diabetes.ipynb
```

Required Deliverables

1. All 8 Jupyter notebooks or Python files
2. One PDF document with all outputs and visualizations
3. A README.txt file explaining:
    - How to run the code
    - Any additional libraries used
    - Kaggle dataset download instructions

Dataset Download Instructions

```python
# Install Kaggle API
!pip install kaggle


# Place your kaggle.json in ~/.kaggle/
# Download datasets using:
!kaggle datasets download -d <dataset-path>
```

Evaluation Rubric

Criteria	Excellent (90-100%)	Good (75-89%)	Satisfactory (60-74%)	Needs Improvement (<60%)
Code Functionality	Code runs perfectly, handles edge cases	Code runs with minor issues	Code has some errors	Code doesn't run
Data Preprocessing	Comprehensive preprocessing, proper handling of missing values	Good preprocessing with minor gaps	Basic preprocessing	Poor or no preprocessing
Model Implementation	Correct implementation with optimal parameters	Correct implementation, suboptimal parameters	Basic implementation	Incorrect implementation
Evaluation	All required metrics with interpretation	Most metrics included	Some metrics missing	Poor or no evaluation
Code Quality	Well-commented, organized, efficient	Good structure, some comments	Basic structure	Poor organization
Visualization	Clear, informative, professional	Good visualizations	Basic plots	Poor or missing

---

Bonus Section (Optional - 10 Extra Marks)

Bonus Question: Model Comparison Dashboard

Dataset: Use any ONE dataset from above
Marks: 10

Task: Create a comprehensive comparison of all four algorithms on the same dataset.

Requirements:

1. Choose one dataset (e.g., Titanic or Diabetes)
2. Train all 4 models (Linear/Logistic Regression, Decision Tree, Random Forest, KNN)
3. Create a comparison table showing:
    - Training time
    - Prediction time

- Accuracy/R^2 Score
- Precision, Recall, F1 (for classification)
- Memory usage

4. Create visualizations:
    - Bar chart comparing accuracies
    - ROC curves for all models on same plot (classification)
    - Feature importance comparison (where applicable)
5. Write a 200-word analysis recommending the best model

Expected Output:

```
Model Comparison on [Dataset Name]:

Performance Metrics:
 Model              | Accuracy | Precision | Recall | F1-Score | Train Time
 -------------------|----------|-----------|--------|----------|------------
 Logistic Regression| X.XXXX   | X.XXXX    | X.XXXX | X.XXXX   | X.XXs
 Decision Tree      | X.XXXX   | X.XXXX    | X.XXXX | X.XXXX   | X.XXs
 Random Forest      | X.XXXX   | X.XXXX    | X.XXXX | X.XXXX   | X.XXs
 KNN                | X.XXXX   | X.XXXX    | X.XXXX | X.XXXX   | X.XXs

Recommendation:
 [Your 200-word analysis here]
```

Notes for Students

1. **Feature Scaling**: Remember, KNN REQUIRES feature scaling, while Decision Trees do NOT
2. **Random State**: Always use random_state=42 for reproducibility
3. **Cross-Validation**: Use for K value selection in KNN and hyperparameter tuning
4. **Categorical Encoding**: Use Label Encoding or One-Hot Encoding as appropriate
5. **Missing Values**: Handle before training
6. **Imbalanced Data**: Check class distribution and handle if needed
7. **Visualization**: Use matplotlib and seaborn for professional plots
8. **Comments**: Explain your preprocessing steps and model choices

---

Sample Code Template

python

```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import *
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
df = pd.read_csv('dataset.csv')

# Explore data
print(df.head())
print(df.info())
print(df.describe())
print(df.isnull().sum())

# Preprocessing
# ... your code here ...

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Feature scaling (if needed)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Model training
model = YourModel()
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)

# Evaluation
```

```
# ... your metrics here ...

# Visualization
# ... your plots here ...
```

Good Luck! 🎯

Remember:

- Read each question carefully
- Follow the requirements exactly
- Test your code before submission
- Comment your code
- Handle errors gracefully