Report for Machine Learning

Ruoxin Chen

Clustering

Report Task 1:

I applied K-means and Gaussian Mixture Model (GMM) clustering algorithms on a 10,000-point subset of the Covertype dataset without using class labels. I set the number of clusters to (K = 7) as specified.

- **K-means**: Used default parameters with 'k-means++' initialization for efficient centroid placement. Random state was fixed for reproducibility.
- **GMM**: Employed default settings with 'full' covariance type to allow each component its own general covariance. Random state was fixed.

Without access to class labels, we relied on default settings to prevent bias and overfitting, ensuring a fair comparison based solely on data features.

Report Task 2:

- **Total pairs with the same class label**: **18,777,683**
- **K-means**: **15,349,480 errors** (Accuracy: 18.26%)
- **GMM**: **11,946,650 errors** (Accuracy: 36.38%)
- **Random Baseline**: **16,095,675 errors** (Accuracy: 14.28%)

Report Task 3:

- **GMM** outperformed K-means with fewer errors and higher accuracy. This suggests that the data clusters may not be spherical and are better captured by the Gaussian assumptions of GMM, which models clusters with varying shapes and covariances.
- **K-means** had more errors, indicating that its assumption of equal variance and spherical clusters doesn't align well with the actual data distribution in this case.
- **Random Baseline** performed worst that highlights the effectiveness of both clustering algorithms over random assignment.

Classification

Report Task 4

Training an SVM on the Covertype dataset is challenging due to the dataset's size and complexity. SVMs can be slow and memory-intensive on large datasets, making them harder to train efficiently. I trained three other methods in less than two minutes, while training SVM took half an hour to complete, which was very **time-consuming**.

Report Task 5

1. **Logistic Regression**: The default number of iterations(100) may not be sufficient for convergence especially with the Covertype dataset, so I used **max_iter=1000** to ensure convergence, with standard scaling to support gradient-based optimization.
2. **Decision Tree**: Applied default settings as decision trees handle raw data well and don't require scaling. **Preprocessing**(Rejected Approach): I tried to scale the data, but it did not provide any improvement, as Decision Trees are insensitive to feature scaling.
3. **Random Forest**: Used default settings(**n_estimators=100**). Random Forest averages predictions from multiple trees, so it performs well without extensive parameter tuning. Default Random Forest setting also uses bagging(Bootstrap Aggregating) to create multiple decision trees, and each tree is trained on a different random subset of data and features, so it helps to capture complex patterns effectively while reducing variance and overfitting.

All algorithms are set with **a fixed random_state** for reproducibility

Report Task 6

- **Logistic Regression (72.47%)**: It exhibits lower accuracy compared to the other models because it assumes a linear decision boundary. This limitation means logistic regression struggles with complex patterns.
- **Decision Tree (93.87%)**: It performs good on training data, and captures nonlinear relationships well but a little overfit as a single model.
- **Random Forest (95.50%)**: It performes the best due to ensemble averaging, which improves robustness and reduces overfitting.

Summary Table

Model	**Accuracy**	**Strengths**	**Weaknesses**
Logistic Regression	72.47%	Simple, fast, interpretable	Limited by linear assumptions; struggles with complexity
Decision Tree	93.87%	Captures nonlinear relationships; no preprocessing	Prone to overfitting as a single model (particularly when the tree depth is unrestricted.)
Random Forest	95.50%	Suitable for high-precision scenes	More resource-intensive than single models

Regression

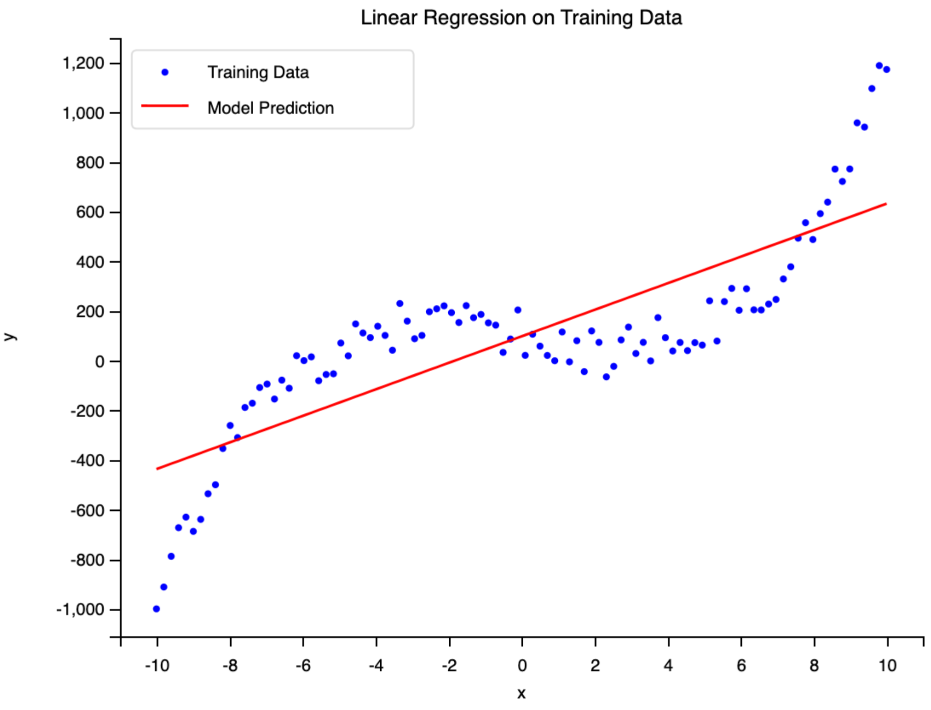
Report Task 7

1. **Linear Regression**:
   - **Choice of Model**: Linear regression was chosen as a baseline model due to its simplicity and interpretability. Linear models provide an initial benchmark and are well-suited for data with linear relationships.

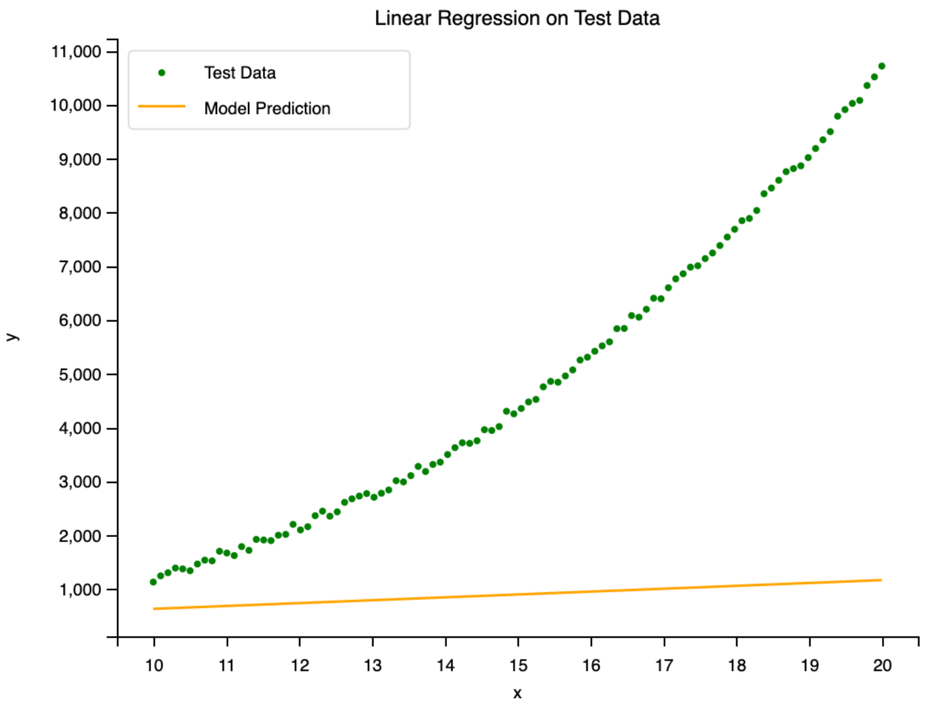
2. **Neural Network**:
    - **Architecture Selection**: A simple neural network with three hidden layers (128, 64, and 32 units) was selected. The neural network was designed to capture non-linear patterns in the data that the linear model could not. This architecture was determined through testing with different layer sizes and configurations, balancing complexity and performance.
    - **Hyperparameters**: A learning rate of 0.01 and 10000 epochs were chosen. This combination allowed the model to converge effectively, reducing training loss while avoiding overfitting.
    - **Optimizer and Activation Function**: The Adam optimizer was chosen for its efficiency, and the ReLU activation function was applied to each hidden layer to introduce non-linearity.
3. **Bayesian Method**:
    - **Parameter Priors**: I used Normal priors with a mean of 0 and a standard deviation of 10 for `alpha` and `beta`, and a HalfNormal prior with a standard deviation of 1 for `sigma`. This approach balances flexibility and constraint based on empirical observations of the data distribution.
    - **Sampling and Posterior Predictive Check**: Sampling 2,000 posterior samples with 1,000 tuning steps allowed for a robust posterior estimation while balancing runtime. These choices were informed by the need for accuracy in posterior predictions and convergence in Bayesian estimation.

Report Task 8:

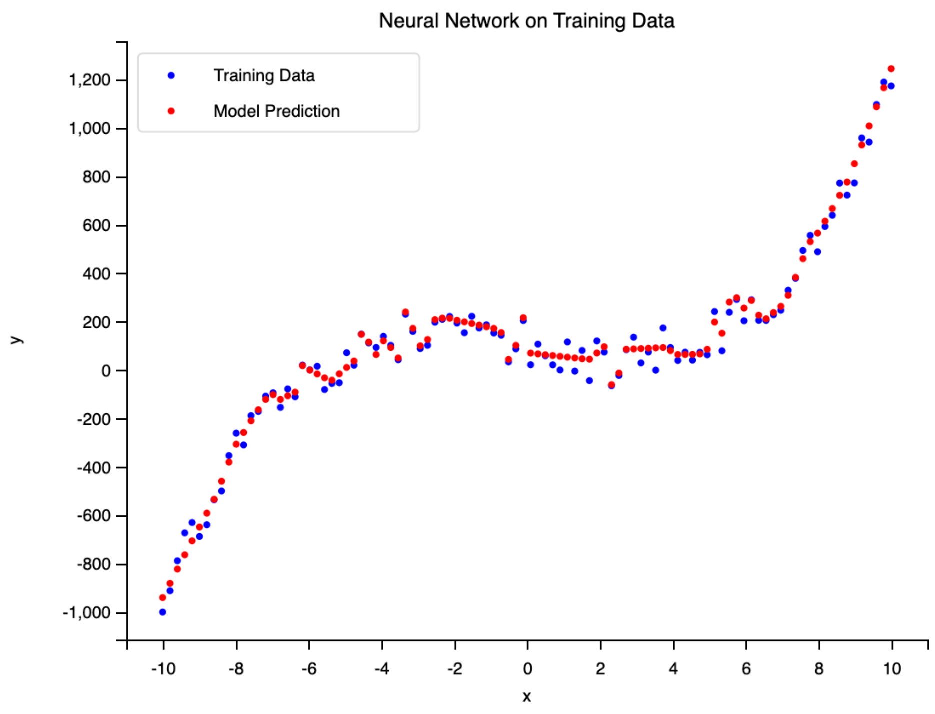
1. **Test Set Mean Squared Errors**:
    - Linear Regression MSE: **22848693**
    - Neural Network MSE: **3088130**
2. **Figures**:
    - **Linear Regression Figures**:
        - **Figure 1**: Scatter plot of training data with linear regression predictions. The plot reveals a poor fit on the training set, as the linear model fails to capture non-linear patterns.



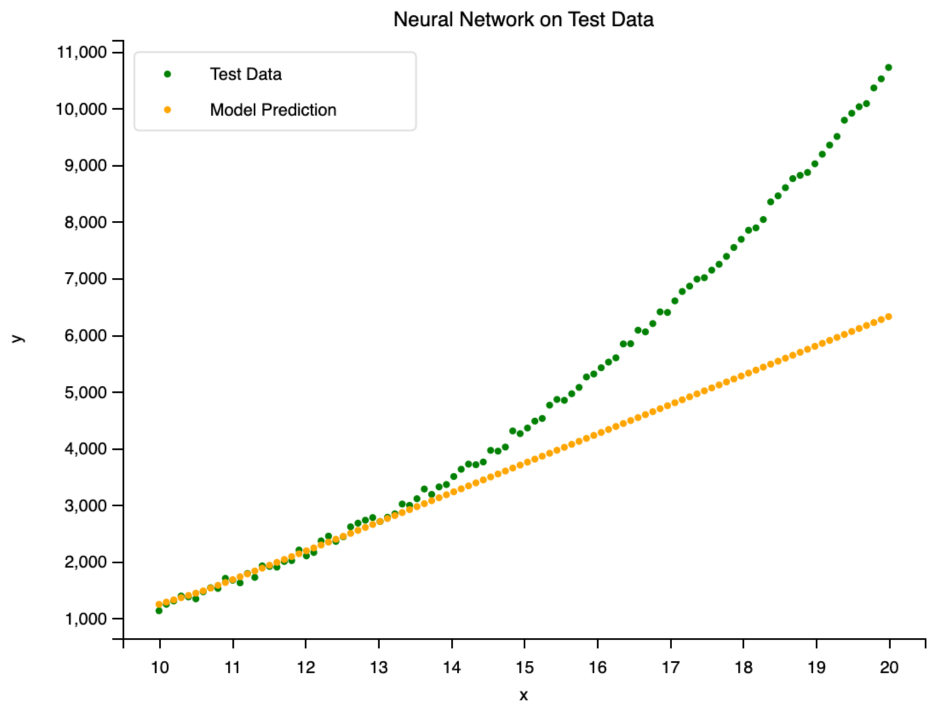
- **Figure 2**: Scatter plot of test data with linear regression predictions. The model performs poorly on the test set, with high MSE, confirming that the data has non-linear characteristics.



- **Neural Network Figures**:
  - **Figure 3**: Scatter plot of training data with neural network predictions. The neural network closely matches the training data points, capturing the non-linear trend effectively.



- **Figure 4**: Scatter plot of test data with neural network predictions. The model shows a good fit on the test set, with significantly lower MSE compared to linear regression, indicating successful generalization.



Report Task 9:

The **linear regression model** achieved an MSE of **22848693** on the test set, indicating poor performance. The model's linear nature makes it unsuitable for capturing the complex, non-linear patterns evident in the data.

The **neural network** significantly outperformed the linear regression model, achieving a test set MSE of **3088130**. The network's multi-layer architecture allowed it to capture non-linear relationships, yielding predictions that closely matched both the training and test data. This result highlights the effectiveness of neural networks in handling complex datasets with non-linear characteristics.

Hidden Markov Models

Report Task 10:

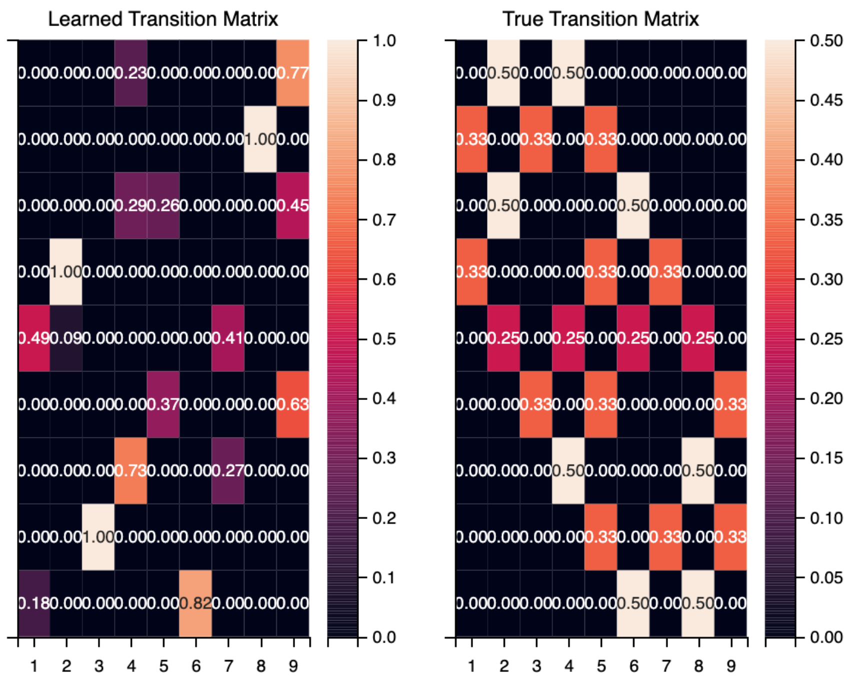
1. Initial Probabilities

The initial state probabilities, when estimated without using transition probabilities, are heavily concentrated on a single state (`[0, 0, 0, 0, 0, 0, 1, 0, 0]`). This diverges considerably from the initial probabilities estimated by the model with transition probabilities (`[0, 0, 0.873, 0, 0, 0, 0.003, 0, 0.124]`). This concentration suggests that without transition information, the EM algorithm struggles to accurately represent the likelihood of beginning in various states.

2. Transition Probability Matrix

Significant deviations appear in the transition matrix when estimated without transition probabilities, showing an average error of `0.173` and a maximum error of `1.0`. Key discrepancies include:

- The transition from State 2 to State 8, estimated at `1.0` (compared to the true value `0`), and the transition from State 1 to State 9, estimated at `0.769` (compared to `0`). These errors reflect challenges in capturing transition dynamics accurately without initial guidance on transitions.



3. Emission Probability Matrix

The emission probabilities also show notable discrepancies between the two methods, with some states either over- or underestimating emissions:

- For example, State 1's learned emissions `[0, 0, 1]` differ from the estimates `[0.095, 0.905, 0]` provided by the model with transition probabilities. These differences indicate that the emission estimates become less accurate when transition probabilities are not included in the model's initial setup.