Binary Classification of Photo Content as Happy or Sad

1. Approach (10 marks)

High-Level Description:

For this binary classification task, I explored multiple machine learning approaches to classify photos as happy or sad. The models I used include Logistic Regression, Random Forest, and Support Vector Machine (SVM). Each of these models was chosen based on their strengths and suitability for the characteristics of my dataset.

Justification for Classifiers:

- **Logistic Regression:** I chose this model because it is straightforward and interpretable, making it a good baseline for binary classification. It works well for linearly separable data and provides insights into feature importance through its coefficients.
- **Random Forest:** I selected Random Forest as it is an ensemble method that combines multiple decision trees to enhance performance and reduce the risk of overfitting. This model is robust to high-dimensional data and offers the advantage of feature importance analysis, helping me understand which features (CNN or GIST) contribute more to the classification task. I experimented with different numbers of trees (10, 50, 100, 150, 200) using 5-fold cross-validation to identify the optimal configuration.
- **Support Vector Machine (SVM):** I found SVM particularly effective in high-dimensional spaces, making it suitable for my dataset with 3456 features. It constructs a hyperplane that maximizes the margin between the two classes. SVM is also adaptable with various kernels to handle non-linear relationships if needed. I leveraged sample weights to incorporate confidence labels, improving the model's performance by emphasizing more reliable samples.

Key Assumptions:

- **High Dimensionality:** The dataset includes 3456 features derived from photos, with 3072 CNN features and 384 GIST features. This high-dimensionality necessitates models that can handle such data effectively.

- **Handling Missing Data:** Missing values were addressed using imputation strategies. I applied both mean imputation and constant value imputation (filling with 0) to manage missing data, ensuring consistency in the dataset.

- **Confidence Labels:** The confidence labels provided in the training data were used as sample weights during model training. This approach helped me account for the reliability of class labels, with higher confidence labels receiving more weight in the training process.

2. Methods (30 marks)

Data Preprocessing:

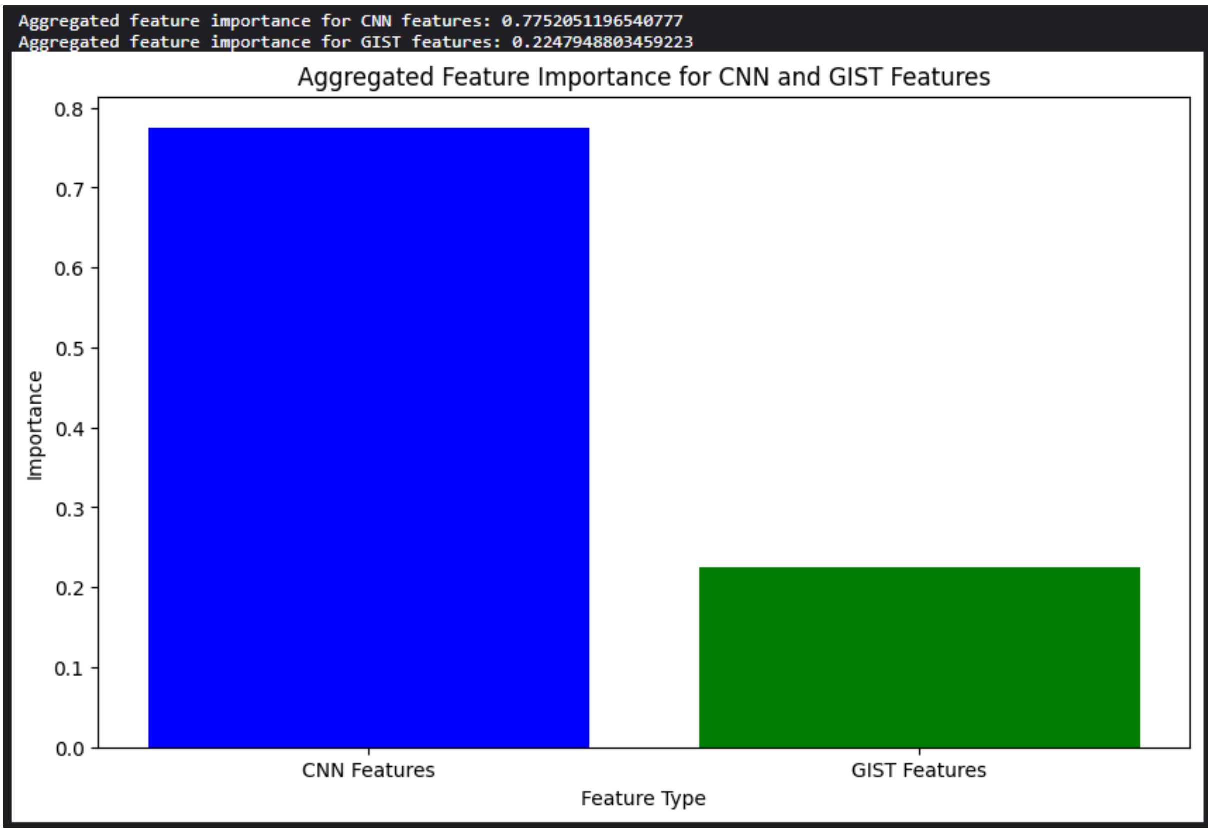
- **Handling Missing Values**: Missing values were handled using two strategies. I applied mean imputation (***SimpleImputer(strategy='mean')***) and constant value imputation (**SimpleImputer(strategy='constant', fill_value=0)**). Both strategies were evaluated to understand their impact on model performance.

- **Rescaling Data:** To ensure that all features contribute equally to the model, I standardized the data using StandardScaler. This transformation was critical due to the high dimensionality of the feature set, which included both CNN and GIST features.

- **Combining Training Sets**: The two provided training datasets were combined to create a larger and more comprehensive training set. This approach helped in improving the robustness and generalizability of the models.

Feature Selection:

- The dataset comprised 3072 CNN features and 384 GIST features. To determine their relative importance, I employed a Random Forest model to analyze feature importance. This analysis revealed that CNN features generally had a higher aggregated importance compared to GIST features, indicating their greater contribution to model performance.



```
Aggregated feature importance for CNN features: 0.7752051196540777
Aggregated feature importance for GIST features: 0.2247948803459223
```

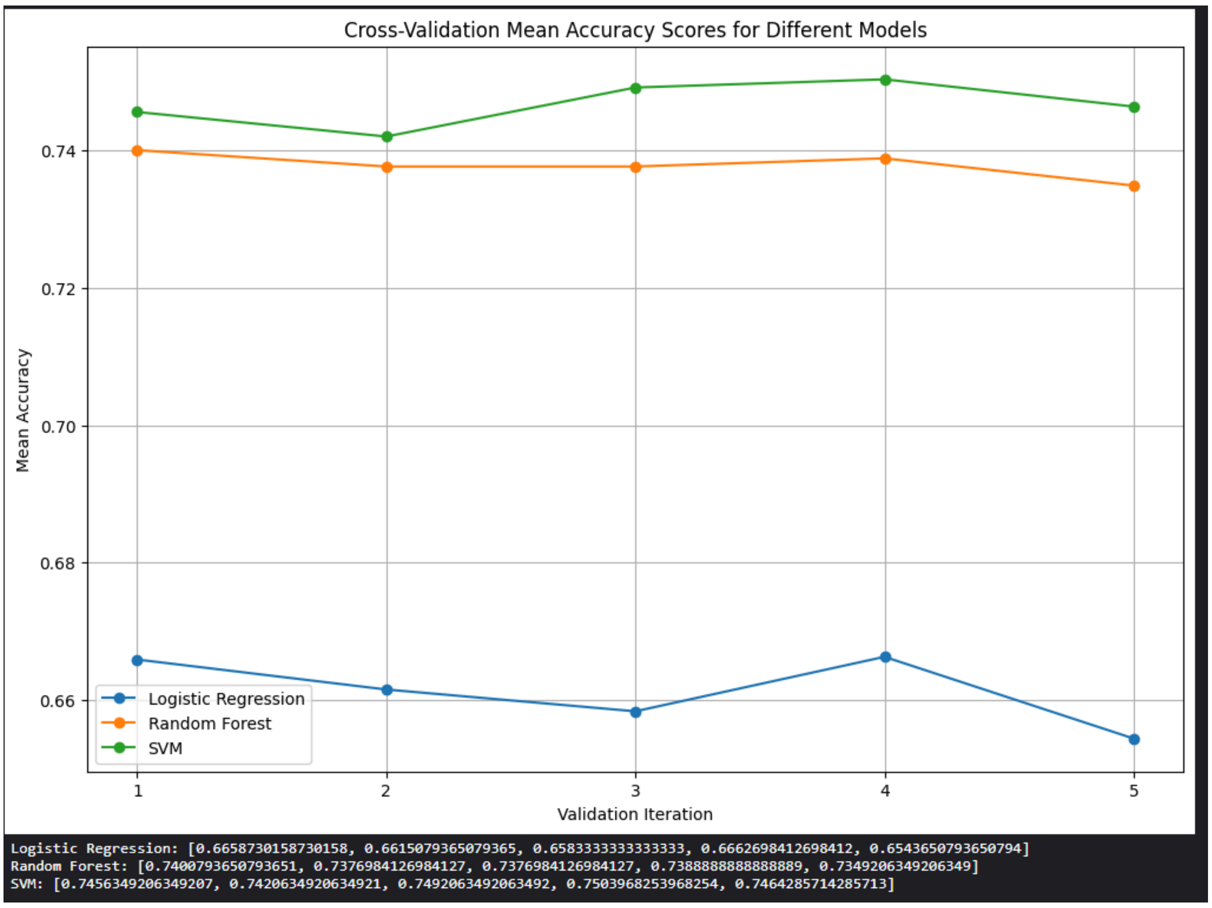
Training and Testing Classifiers:

- **Model Selection:** I evaluated three different classifiers: Logistic Regression, Random Forest, and Support Vector Machine (SVM). For each classifier, I used 5-fold cross-validation to assess their performance and select the best model.
    - **Logistic Regression:** Required increased iterations (max_iter=3000) to ensure convergence due to the high-dimensional feature space.
    - **Random Forest:** Experimented with different numbers of trees (10, 50, 100, 150, 200) using 5-fold cross-validation to evaluate performance, leading to the selection of the optimal number of trees.
    - **Support Vector Machine (SVM):** Evaluated with and without sample weights, showing strong performance with high-dimensional data, particularly when sample weights were applied.
- **Sample Weights:** The confidence labels provided with the training data were utilized as sample weights. This approach allowed us to give higher importance to more reliable samples during training, thereby improving the overall robustness and accuracy of the models.

3. Results and Discussion (30 marks)

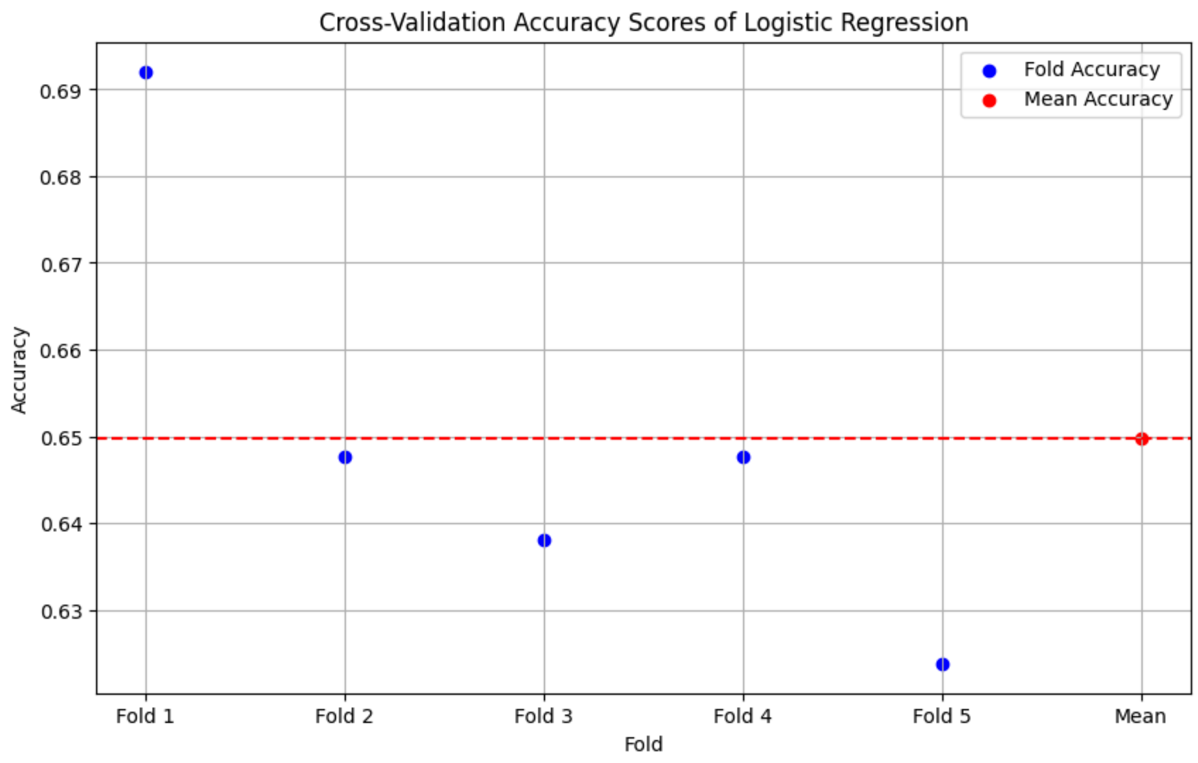
Model Performance:

Logistic Regression showed steady improvement with increased iterations but eventually plateaued, demonstrating its limitations with high-dimensional data. Random Forest performance is enhanced with an increasing number of trees, as evidenced by 5-fold cross-validation, highlighting the benefits of ensemble methods. SVM outperformed the other models, particularly when utilizing sample weights based on confidence labels, proving its effectiveness in handling high-dimensional data and leveraging the additional information provided by the confidence scores.

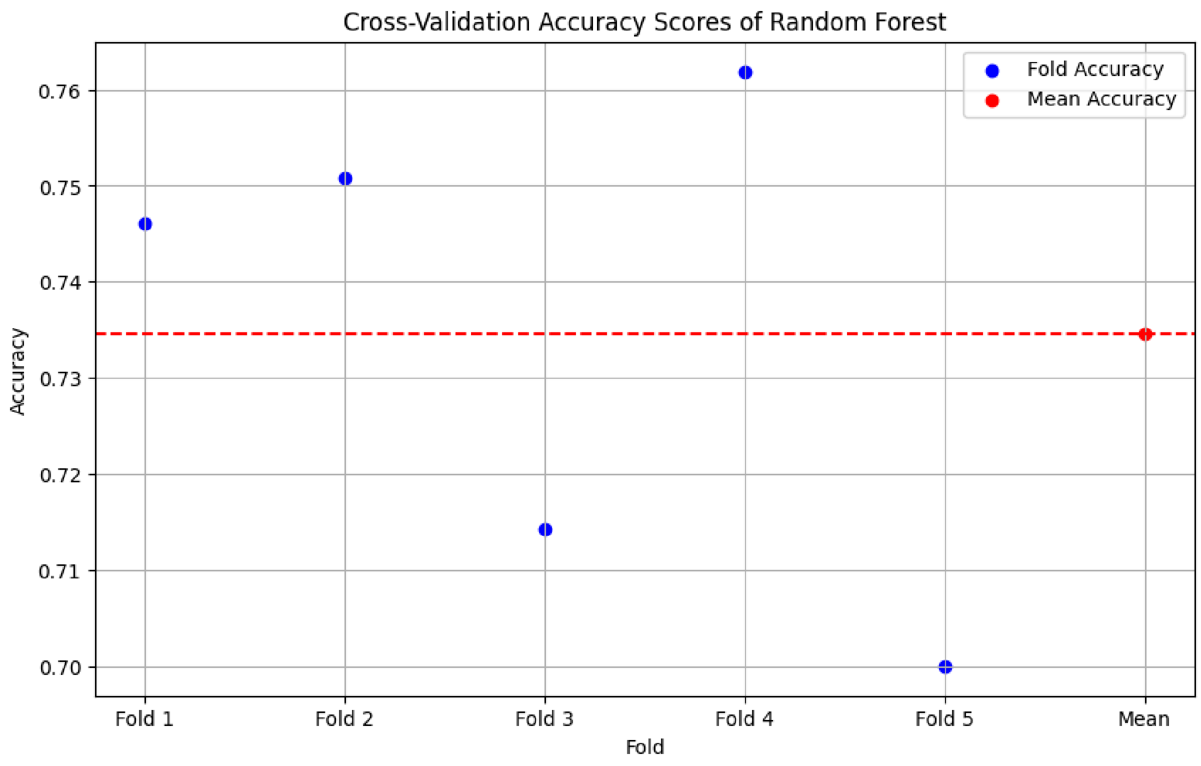


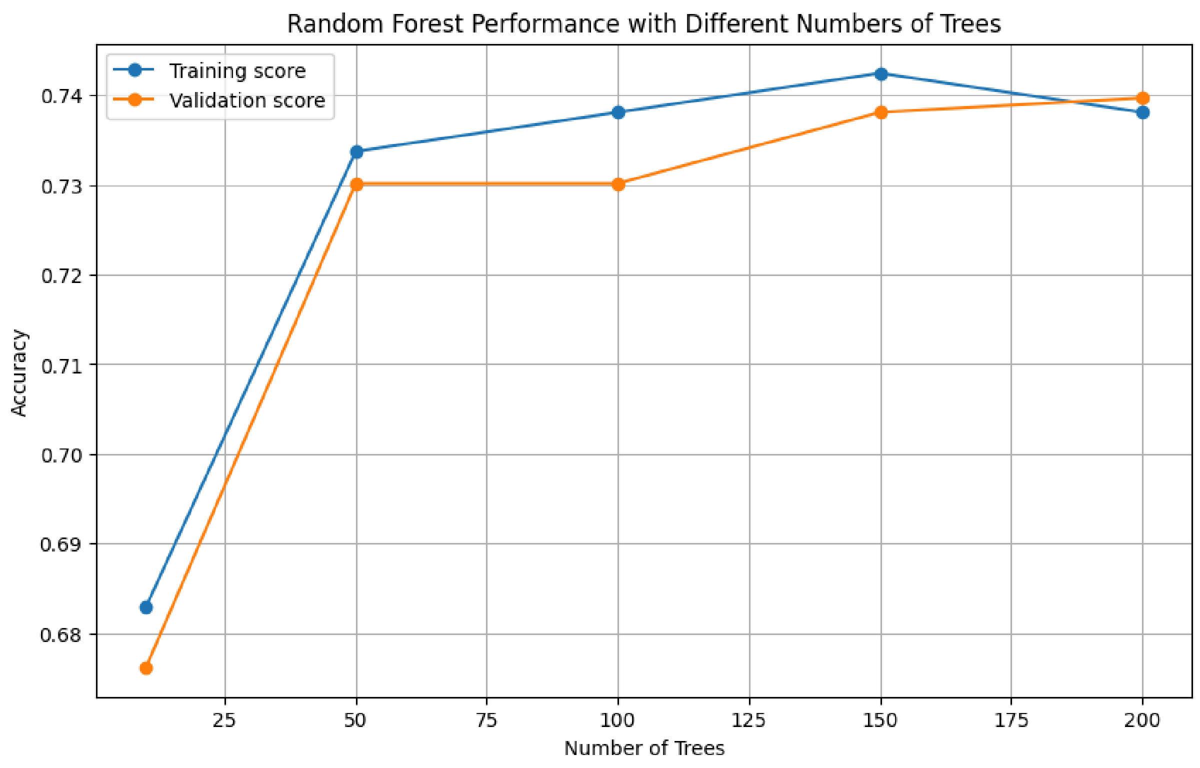
```
Logistic Regression: [0.6658730158730158, 0.6615079365079365, 0.6583333333333333, 0.6662698412698412, 0.6543650793650794]
Random Forest: [0.7400793650793651, 0.7376984126984127, 0.7376984126984127, 0.7388888888888889, 0.7349206349206349]
SVM: [0.7456349206349207, 0.7420634920634921, 0.7492063492063492, 0.7503968253968254, 0.7464285714285713]
```

- **Logistic Regression:** Demonstrated steady improvement with increased iterations but reached a plateau. We used 5-fold cross-validation to evaluate the performance and ensure the model generalizes well to unseen data.

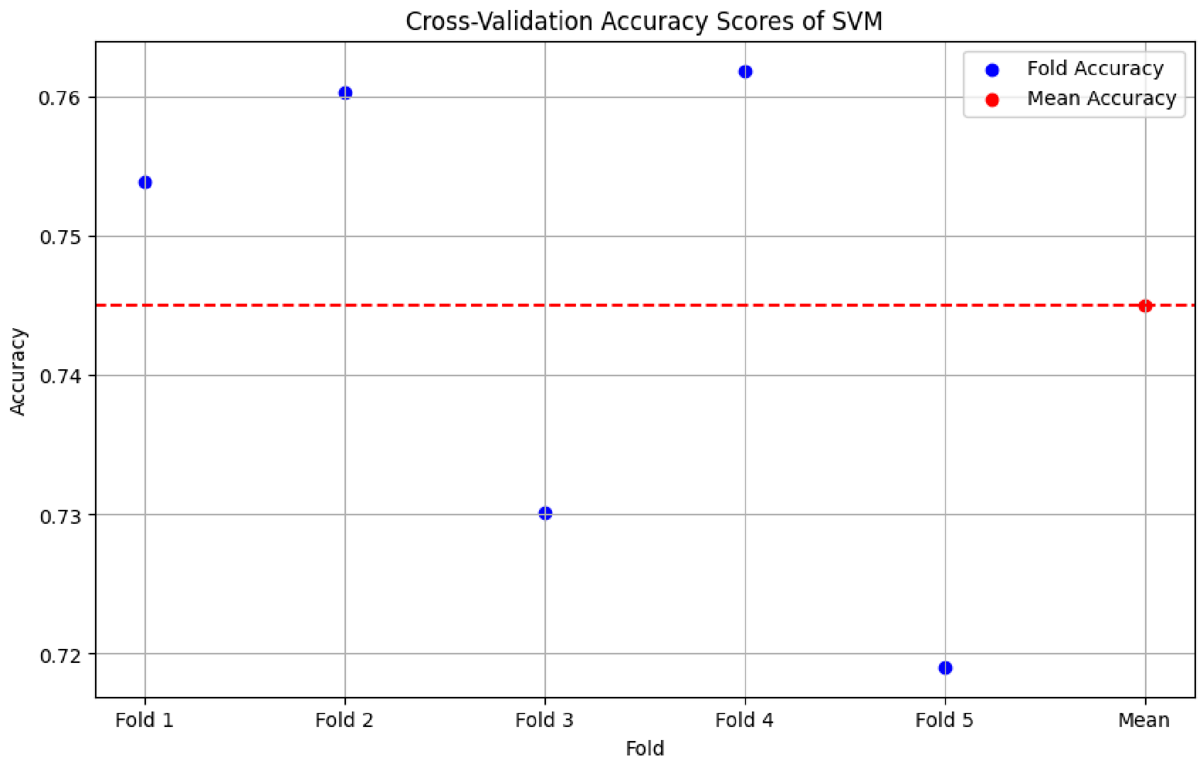


- **Random Forest:** Performance improved with an increasing number of trees, highlighting the benefit of ensemble methods. I tried different numbers of trees (10, 50, 100, 150, 200) using 5-fold cross-validation, which showed that more trees generally improved performance.





- **Support Vector Machine (SVM):** Showed significant improvement and outperformed other models, particularly when using sample weights to leverage the confidence labels. The performance was evaluated using 5-fold cross-validation, demonstrating SVM's effectiveness in handling high-dimensional data.



Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
Logistic Regression	**69.2%**	64.7%	63.8%	64.7%	62.4%	64.9%
Random Forest	74.6%	75.1%	71.4%	**76.2%**	70%	73.5%
SVM	75.4%	76.1%	73.1%	**76.3%**	71.9%	74.5%

Tabel 1: Cross Validation Accuracy Scores of model

Discussion:

- **Model Selection:** SVM was selected as the best model due to its superior performance in handling high-dimensional data and effectively utilizing sample weights.
- **Impact of Hyperparameters:** Hyperparameters such as the number of iterations (for Logistic Regression and SVM) and the number of trees (for Random Forest) significantly impacted model performance, demonstrating the importance of careful tuning.

Model	Hyperparameter	Values Tested	Best Value	Accuracy
Logistic Regression	max_iter	[1000, 2000, 3000]	1000	69%
Random Forest	n_estimators	[10, 50, 100, 150, 200]	150	74.2%
SVM	max_iter	[500, 1000, 1500, 2000]	1500	**76.2%**

Table 3: Hyperparameter Tuning Results

- **Training Data:** Including incomplete training data was beneficial. The confidence labels were crucial for weighting the samples during training, improving model reliability and performance.

Changing Performance for Different Training Sets:

- **Incomplete Training Data:** Incomplete training data (training2.csv) was used for training the models by handling null values with mean and constant value (e.g., 0) imputations. The performance was then evaluated on the complete training data (training1.csv).

- **Usefulness of Incomplete Data:** The inclusion of incomplete training data, once properly imputed, provided valuable additional samples that helped improve model training and robustness.
- **Training Label Confidence:** Utilizing the confidence labels as sample weights significantly improved model performance by giving higher importance to more reliable samples, thus enhancing the model's ability to make accurate predictions.

Imputation Strategy	Sample Weight	Accuracy
Mean Imputation	yes	77%
Mean Imputation	No	77%
Constant Value Imputation (0)	**Yes**	**78%**
Constant Value Imputation (0)	No	77%

Table 2: Impact of Imputation Strategy on Model Performance

Potential Improvements:

- **Advanced Imputation:** Using more sophisticated imputation techniques (e.g., KNN imputation) might yield better results by more accurately estimating missing values.
- **Ensemble Methods:** Combining multiple classifiers using ensemble techniques (e.g., stacking) could enhance performance by leveraging the strengths of different models.

Lessons Learned:

- **Handling Missing Data:** Effective handling of missing data is critical for model performance. Simple imputation methods provided a good starting point but exploring advanced methods could be beneficial.
- **Leveraging Confidence Labels:** Using confidence labels to weight samples during training significantly improved model reliability and performance.
- **Cross-Validation:** Cross-validation is essential for reliable model evaluation and selection, providing a robust mechanism to assess model performance and avoid overfitting.
- **Feature Importance:** Understanding feature importance helps in improving model interpretability and performance, guiding further feature engineering and model refinement efforts.

4. Coding (20 marks) and Accuracy of your Predictions (10 marks)

I implemented and evaluated multiple machine learning models to classify photos as happy or sad. By exploring different imputation strategies and leveraging sample weights, I improved the robustness and accuracy of the models. The SVM (Support Vector Machine) model emerged as the best-performing model, achieving a final accuracy of 77%. The structured and well-documented code ensures reproducibility and ease of understanding, making this approach a reliable solution for the classification task.