Performance & Final Submission Phase

Solution Performance

Project Development Phase
Model Performance Test

Performance Testing – Artificial Intelligence

Date	19 November 2023
Team ID	Team-591988
Project Name	Project - AI Body Language Detector Using Media pipe
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot

1.	Model Summary		]: # Model Summary
			rf_model = fit_models['rf']
			
			# Access the RandomForestClassifier from the pipeline
			rf_classifier = rf_model.named_steps['randomforestclassifier']
			
			# Get the number of trees (estimators) in the forest
			n_estimators = rf_classifier.n_estimators
			print(f"Number of Estimators (Trees): {n_estimators}")
			
			# Get feature importances (if applicable)
			feature_importances = rf_classifier.feature_importances_
			print("Feature Importances:")
			for feature, importance in zip(X.columns, feature_importances):
			    print(f"{feature}: {importance}")
			
			Number of Estimators (Trees): 100
			Feature Importances:
			0: 0.0
			1: 4.3254354878427415e-05
			2: 0.002511131129914398
			3: 0.002640077501274246
			4: 0.0
			5: 0.0
			6: 0.0
			7: 0.0010368137421092 47
			8: 0.00022671438726009802
			9: 0.0002297972998864533
			10: 0.0019030727023623165
			11: 0.002945873278431087
			12: 0.0
			13: 5.319454118895791e-05
			14: 0.00015355536107018465
			15: 0.006268527477386391

2.	Accuracy	Training Accuracy = 1.0
		
		Validation Accuracy = 0.98
			In [67]: #training accuracy
			rf_model = fit_models['rf']
			
			# Predictions on the training data
			y_train_pred = rf_model.predict(X_train)
			
			# Calculate training accuracy
			training_accuracy = accuracy_score(y_train, y_train_pred)
			print(f"Training Accuracy: {training_accuracy}")
			
			Training Accuracy: 1.0

			In [69]: <pre>from sklearn.model_selection import train_test_split
			from sklearn.metrics import accuracy_score
			
			# Assuming you have a DataFrame named df with your data
			
			# Split the data into features (X) and target (y)
			X = df.drop('class', axis=1)
			y = df['class']
			
			# Split the data into training and validation sets
			X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=1234)
			
			# Assuming you have a fitted Random Forest model stored in fit_models['rf']
			rf_model = fit_models['rf']
			
			# Predictions on the validation data
			y_val_pred = rf_model.predict(X_val)
			
			# Calculate validation accuracy
			validation_accuracy = accuracy_score(y_val, y_val_pred)
			print(f"Validation Accuracy: {validation_accuracy}")</pre>
			Validation Accuracy: 0.9815668202764977

Performance Testing – Data Analytics

Model Performance Testing:

<table>
<tr><th>Parameter</th><th>Screenshot / Values</th></tr>
<tr><td>Dashboard design</td><td>No of Visualizations = 144
<pre>
n [70]: import mediapipe as mp
        import cv2

        mp_drawing = mp.solutions.drawing_utils
        mp_holistic = mp.solutions.holistic

        cap = cv2.VideoCapture(0)

        # initialize the model
        with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
            frame_count = 0 # Initialize a frame counter

            while cap.isOpened():
                ret, frame = cap.read()
                frame_count += 1 # Increment the frame counter

                image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                image.flags.writeable = False

                results = holistic.process(image)

                # ... (Your existing code for drawing landmarks and exporting coordinates)

                cv2.imshow("Holistic Model Detections", image)

                if cv2.waitKey(30) & 0xFF == ord('q'):
                    break

            print(f"Number of Visualizations: {frame_count}")

        cap.release()
        cv2.destroyAllWindows()

        Number of Visualizations: 144
</pre></td></tr>
<tr><td>Data Responsiveness</td><td>Data Responsiveness = 8.64
<pre>
mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic

cap = cv2.VideoCapture(0)

initialize the model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    frame_count = 0
    start_time = time.time() # Record start time

    while cap.isOpened():
        ret, frame = cap.read()
        frame_count += 1

        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        results = holistic.process(image)

        # ... (Your existing code for drawing landmarks and exporting coordinates)

        cv2.imshow("Holistic Model Detections", image)

        if cv2.waitKey(30) & 0xFF == ord('q'):
            break

    end_time = time.time() # Record end time
    elapsed_time = end_time - start_time
    fps = frame_count / elapsed_time
    print(f"Data Responsiveness (FPS): {fps:.2f}")

cap.release()
cv2.destroyAllWindows()

Data Responsiveness (FPS): 8.64
</pre></td></tr>
<tr><td>Amount Data to Rendered (DB2 Metrics)</td><td>Frame size (Pixels): 307200
Total number of Landmarks: 501</td></tr>
</table>

	ı [72]: `#Amount Data to Rendered (DB2 Metrics)`
	`#Frame Size (Resolution):`
	`height, width, _ = frame.shape`
	`frame_size = height * width`
	`print(f"Frame Size (Pixels): {frame_size}")`
	
	`Frame Size (Pixels): 307200`
	
	ı [76]: `#Amount Data to Rendered (DB2 Metrics)`
	`#Number of Landmarks:`
	`num_coords = len(results.pose_landmarks.landmark)+len(results.face_landmarks.landmark)`
	`num_coords`
	
	`print(f"Total Number of Landmarks: {total_landmarks}")`
	
	`Total Number of Landmarks: 501`

Performance Testing – Machine Learning

Model Performance Testing:

<table>
<tr><th>S.No.</th><th>Parameter</th><th>Values</th><th>Screenshot</th></tr>
<tr><td>1.</td><td>Metrics</td><td>Regression Model:
lr = 0.9907,
rc = 0.995,
rf = 0.981,
gb = 0.986

Classification Model:
Confusion Matrix,
Accuracy Score = 0.981
&
Classification Report</td><td>

Regression Model:

```
In [50]: for algo, model in fit_models.items():
             yhat = model.predict(X_test)
             print(algo, accuracy_score(y_test, yhat))

         lr 0.9907834101382489
         rc 0.9953917050691244
         rf 0.9815668202764977
         gb 0.9861751152073732
```

Classification model:

```
In [78]: from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
         from sklearn.model_selection import train_test_split
         import pandas as pd

         # Assuming you have a DataFrame df with the columns 'class' and other features
         # Replace df with your actual DataFrame

         # Read the CSV file into a DataFrame
         df = pd.read_csv('coords.csv')

         # Extract features (X) and target labels (y)
         X = df.drop('class', axis=1)
         y = df['class']

         # Split the data into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1234)

         # Assuming you have a fitted Random Forest model stored in fit_models['rf']
         rf_model = fit_models['rf']

         # Predictions on the test data
         y_pred = rf_model.predict(X_test)

         # Confusion Matrix
         conf_matrix = confusion_matrix(y_test, y_pred)
         print("Confusion Matrix:")
         print(conf_matrix)

         # Accuracy Score
         acc_score = accuracy_score(y_test, y_pred)
         print(f"Accuracy Score: {acc_score}")

         # Classification Report
         class_report = classification_report(y_test, y_pred)
         print("Classification Report:")
         print(class_report)
```

```
Confusion Matrix:
[[54  0  0  0]
 [ 0 39  0  0]
 [ 0  1 54  0]
 [ 2  0  1 66]]
Accuracy Score: 0.9815668202764977
Classification Report:
              precision    recall  f1-score   support

       Fight       0.96      1.00      0.98        54
       Happy       0.97      1.00      0.99        39
         Sad       0.98      0.98      0.98        55
  Victorious       1.00      0.96      0.98        69

    accuracy                           0.98       217
   macro avg       0.98      0.98      0.98       217
weighted avg       0.98      0.98      0.98       217
```

</td></tr>
<tr><td>2.</td><td>Tune the Model</td><td>Validation Method = 0.981</td><td>

```
[80]: y_val_pred = model.predict(X_test)
      val_accuracy = accuracy_score(y_test, y_val_pred)
      print(f"Validation Accuracy: {val_accuracy}")

      Validation Accuracy: 0.9815668202764977
```

</td></tr>
</table>

Performance Testing – Cyber Security with AI

Model Performance Testing:

<table>
<tr><th>S.No.</th><th>Parameter</th><th>Values</th><th>Screenshot</th></tr>
<tr><td>1.</td><td>Scanning the target</td><td>Risk factors -</td><td><u>Accuracy and Bias</u>: Potential inaccuracies and biases in interpreting body language, leading to incorrect predictions and biased outcomes.

<u>Privacy Concerns</u>: Capture and processing of sensitive information, risking privacy infringement if not handled properly.

<u>Security Vulnerabilities</u>: Susceptibility to adversarial attacks, where attackers manipulate data to deceive the model.

<u>Ethical Considerations</u>: Adherence to ethical standards, including issues of consent, fairness, and accountability.

<u>Data Quality and Representativeness</u>: Impact of biases in training data on model predictions, emphasizing the importance of high-quality and representative data.</td></tr>
<tr><td>2.</td><td>Gaining access</td><td>Access process -

Vulnerability found -</td><td><u>Access Process:</u>
Input: Capture video input of body movements.
Media Pipe Processing: Use Media Pipe to extract spatial landmarks from the video.
Model Inference: Apply a pre-trained model (e.g., Random Forest) to predict body language.
Prediction: Receive real-time predictions with confidence scores.
Visualization: Overlay predictions on the video for user interaction.
Feedback: Allow user feedback for model improvement.
Post-Processing: Log data for analysis and implement privacy measures.
Security: Ensure secure communication and data protection.
Monitoring: Regularly monitor system performance.
User Education: Guide users on system interaction and limitations.
<u>Vulnerability found:</u>
Privacy: Ensure compliance with privacy regulations and implement anonymization techniques for video data.</td></tr>
</table>

			• Data Security: Implement robust measures to secure sensitive information and prevent unauthorized access. • Model Robustness: Assess the model's resilience to adversarial attacks and its generalization to different scenarios. • Communication Security: Use secure protocols to protect data during network communication. • Dependency Security: Regularly update and monitor dependencies to address potential security vulnerabilities. • User Input Validation: Validate and sanitize user inputs to prevent injection attacks or exploitation. • Access Control: Implement proper access controls to restrict system access based on user roles. • Error Handling: Handle errors gracefully to avoid exposing sensitive information.
3.	Maintaining access - Automation (AI implementation)	AI tools used – Automation implemented -	<u>AI tools used:</u> Media pipe, OpenCV, NumPy, Pandas, scikit-learn, CSV <u>Automation implemented:</u> Real-time Pose Estimation and Landmark Detection, Data Collection and Export, Machine Learning Model Training, Model Evaluation, Real-time Inference, Visualization of Landmarks and Detections
4.	Covering Tracks & Report	Vulnerability risk factors	<u>Vulnerability risk factors:</u> Privacy Concerns, Data Security, Model Fairness and Bias, Adversarial Attacks, Model Robustness, Security of Dependencies, Real-time Inference Risks, Ethical Considerations