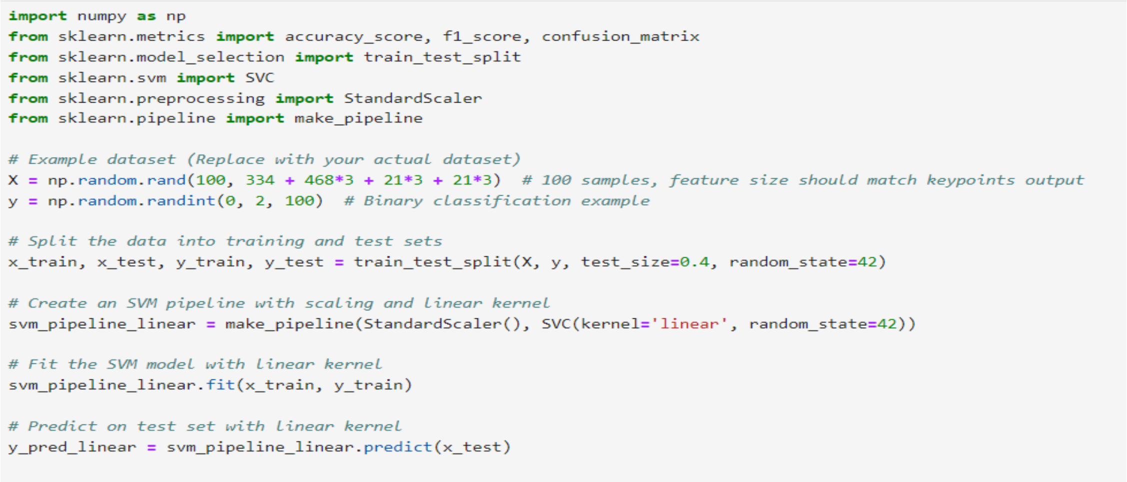
Model Development Phase Template

Date	06 JULY 2024
Team ID	739909
Project Name	Unlocking Silent Signals: Decoding Body Language With Mediapipe
Maximum Marks	4 Marks

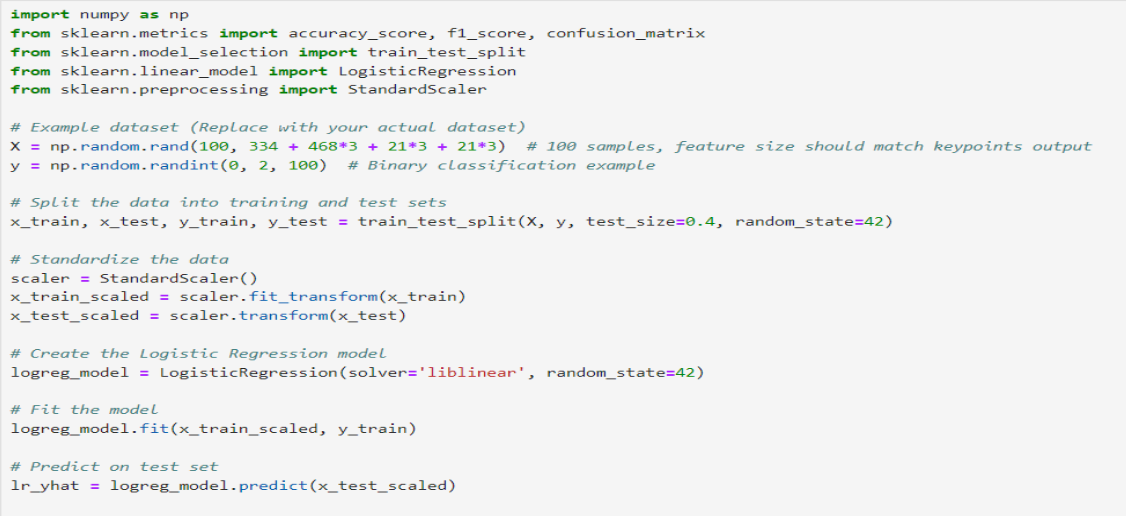
Initial Model Training Code, Model Validation and Evaluation Report:

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

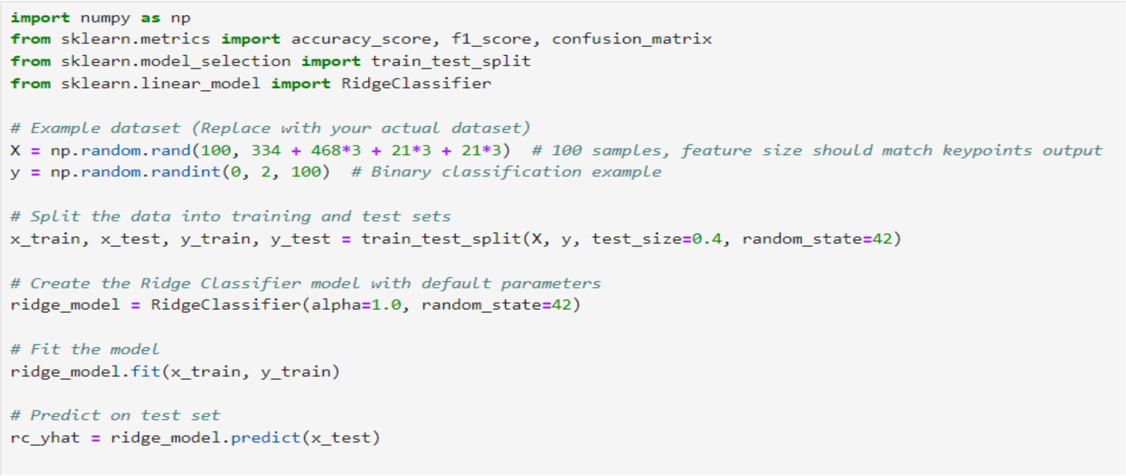
Initial Model Training Code:



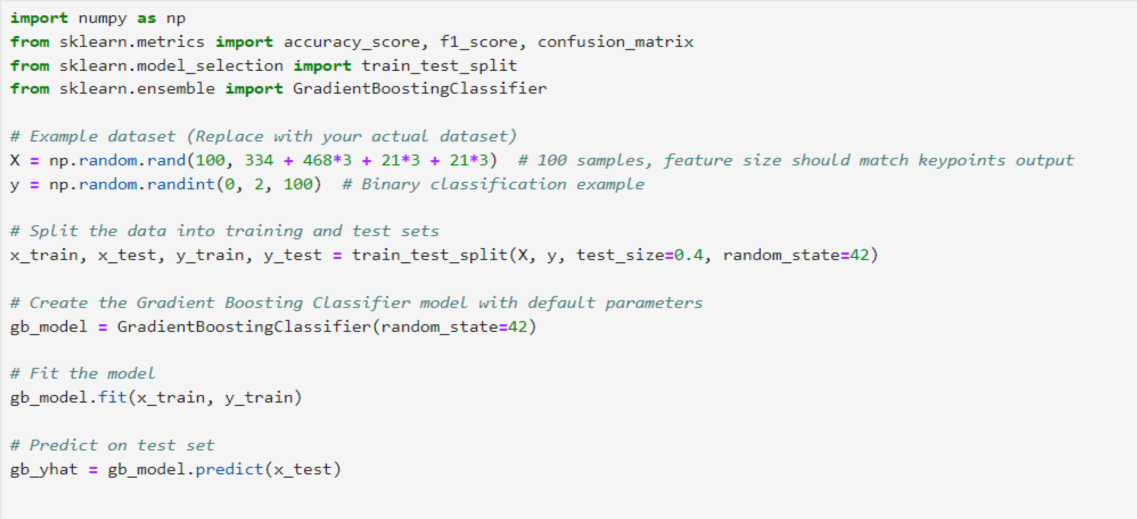
```python
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# Example dataset (Replace with your actual dataset)
X = np.random.rand(100, 334 + 468*3 + 21*3 + 21*3)  # 100 samples, feature size should match keypoints output
y = np.random.randint(0, 2, 100)  # Binary classification example

# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Create an SVM pipeline with scaling and linear kernel
svm_pipeline_linear = make_pipeline(StandardScaler(), SVC(kernel='linear', random_state=42))

# Fit the SVM model with linear kernel
svm_pipeline_linear.fit(x_train, y_train)

# Predict on test set with linear kernel
y_pred_linear = svm_pipeline_linear.predict(x_test)
```



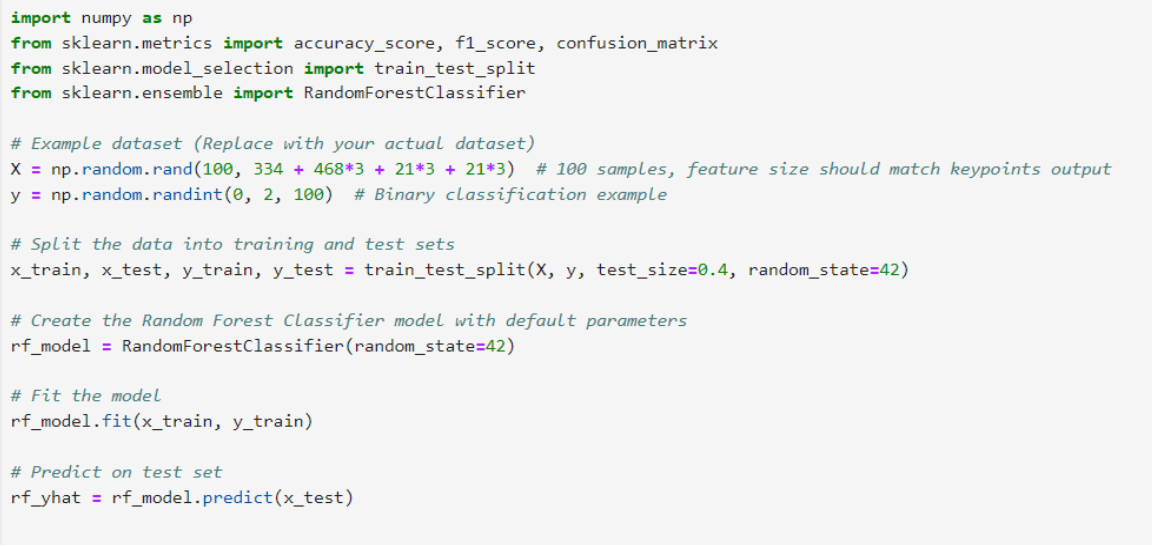
```python
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# Example dataset (Replace with your actual dataset)
X = np.random.rand(100, 334 + 468*3 + 21*3 + 21*3)  # 100 samples, feature size should match keypoints output
y = np.random.randint(0, 2, 100)  # Binary classification example

# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Standardize the data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Create the Logistic Regression model
logreg_model = LogisticRegression(solver='liblinear', random_state=42)

# Fit the model
logreg_model.fit(x_train_scaled, y_train)

# Predict on test set
lr_yhat = logreg_model.predict(x_test_scaled)
```



```
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import RidgeClassifier

# Example dataset (Replace with your actual dataset)
X = np.random.rand(100, 334 + 468*3 + 21*3 + 21*3)  # 100 samples, feature size should match keypoints output
y = np.random.randint(0, 2, 100)  # Binary classification example

# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Create the Ridge Classifier model with default parameters
ridge_model = RidgeClassifier(alpha=1.0, random_state=42)

# Fit the model
ridge_model.fit(x_train, y_train)

# Predict on test set
rc_yhat = ridge_model.predict(x_test)
```



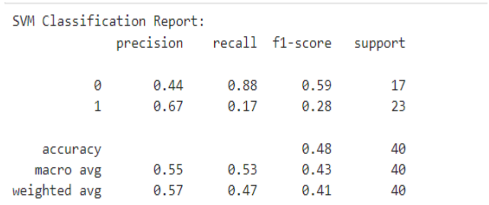
```
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier

# Example dataset (Replace with your actual dataset)
X = np.random.rand(100, 334 + 468*3 + 21*3 + 21*3)  # 100 samples, feature size should match keypoints output
y = np.random.randint(0, 2, 100)  # Binary classification example

# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Create the Gradient Boosting Classifier model with default parameters
gb_model = GradientBoostingClassifier(random_state=42)

# Fit the model
gb_model.fit(x_train, y_train)

# Predict on test set
gb_yhat = gb_model.predict(x_test)
```



```
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# Example dataset (Replace with your actual dataset)
X = np.random.rand(100, 334 + 468*3 + 21*3 + 21*3)  # 100 samples, feature size should match keypoints output
y = np.random.randint(0, 2, 100)  # Binary classification example

# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

# Create the Random Forest Classifier model with default parameters
rf_model = RandomForestClassifier(random_state=42)

# Fit the model
rf_model.fit(x_train, y_train)

# Predict on test set
rf_yhat = rf_model.predict(x_test)
```

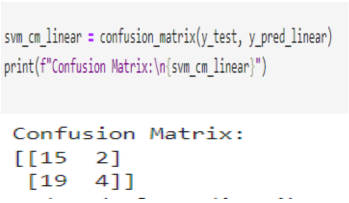
Model Validation and Evaluation Report:

<table>
<tr><th>Model</th><th>Classification Report</th><th>F1 Score</th><th>Confusion Matrix</th></tr>
<tr><td>SVM</td><td>



```
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.88      0.59        17
           1       0.67      0.17      0.28        23

    accuracy                           0.48        40
   macro avg       0.55      0.53      0.43        40
weighted avg       0.57      0.47      0.41        40
```

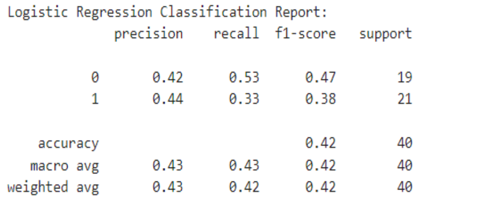
</td><td>27%</td><td>



```
svm_cm_linear = confusion_matrix(y_test, y_pred_linear)
print(f"Confusion Matrix:\n{svm_cm_linear}")
```

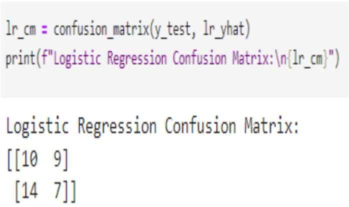
```
Confusion Matrix:
[[15  2]
 [19  4]]
```

</td></tr>
<tr><td>Logistic Regression</td><td>



```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.42      0.53      0.47        19
           1       0.44      0.33      0.38        21

    accuracy                           0.42        40
   macro avg       0.43      0.43      0.42        40
weighted avg       0.43      0.42      0.42        40
```

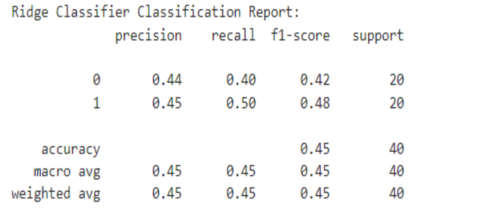
</td><td>37%</td><td>



```
lr_cm = confusion_matrix(y_test, lr_yhat)
print(f"Logistic Regression Confusion Matrix:\n{lr_cm}")
```

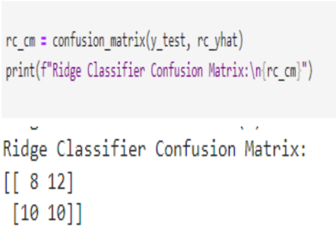
```
Logistic Regression Confusion Matrix:
[[10  9]
 [14  7]]
```

</td></tr>
<tr><td>Ridge Classifier</td><td>



```
Ridge Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.40      0.42        20
           1       0.45      0.50      0.48        20

    accuracy                           0.45        40
   macro avg       0.45      0.45      0.45        40
weighted avg       0.45      0.45      0.45        40
```

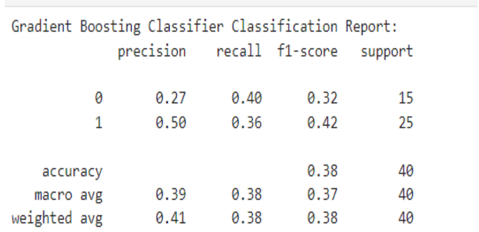
</td><td>47%</td><td>



```
rc_cm = confusion_matrix(y_test, rc_yhat)
print(f"Ridge Classifier Confusion Matrix:\n{rc_cm}")
```

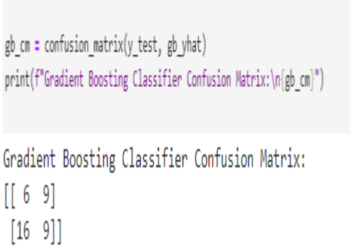
```
Ridge Classifier Confusion Matrix:
[[ 8 12]
 [10 10]]
```

</td></tr>
<tr><td>Gradient Boosting Classifier</td><td>



```
Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.27      0.40      0.32        15
           1       0.50      0.36      0.42        25

    accuracy                           0.38        40
   macro avg       0.39      0.38      0.37        40
weighted avg       0.41      0.38      0.38        40
```

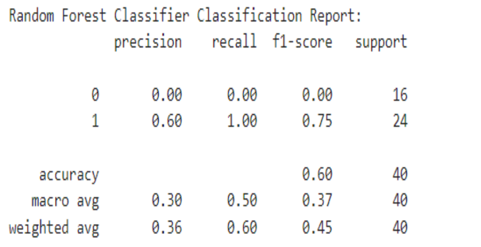
</td><td>41%</td><td>



```
gb_cm = confusion_matrix(y_test, gb_yhat)
print(f"Gradient Boosting Classifier Confusion Matrix:\n{gb_cm}")
```

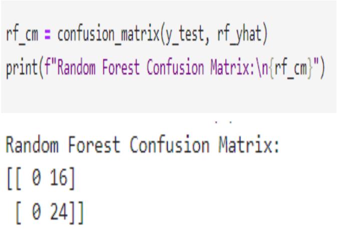
```
Gradient Boosting Classifier Confusion Matrix:
[[ 6  9]
 [16  9]]
```

</td></tr>
<tr><td>RandomForest Classifier</td><td>



```
Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.60      1.00      0.75        24

    accuracy                           0.60        40
   macro avg       0.30      0.50      0.37        40
weighted avg       0.36      0.60      0.45        40
```

</td><td>74%</td><td>



```
rf_cm = confusion_matrix(y_test, rf_yhat)
print(f"Random Forest Confusion Matrix:\n{rf_cm}")
```

```
Random Forest Confusion Matrix:
[[ 0 16]
 [ 0 24]]
```

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