Model Development Phase Template

Date	21 July 2024
Team ID	Team-740025
Project Title	Unlocking Silent Signals :Decoding Body Language with Mediapipe
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

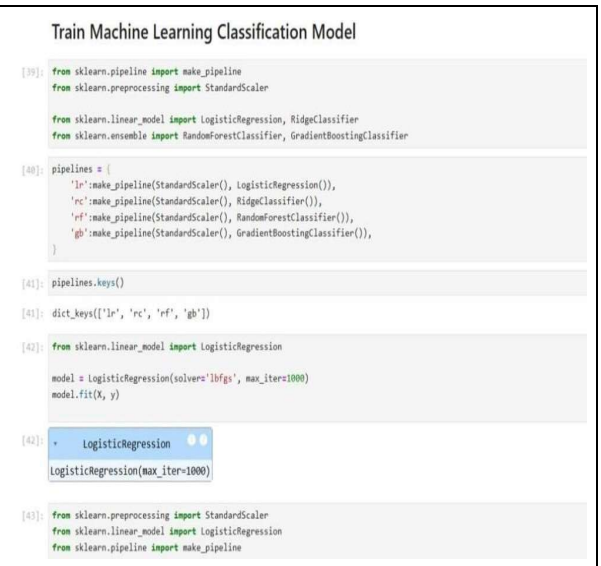
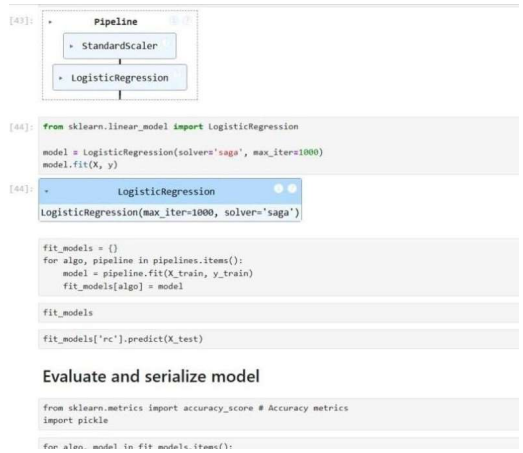
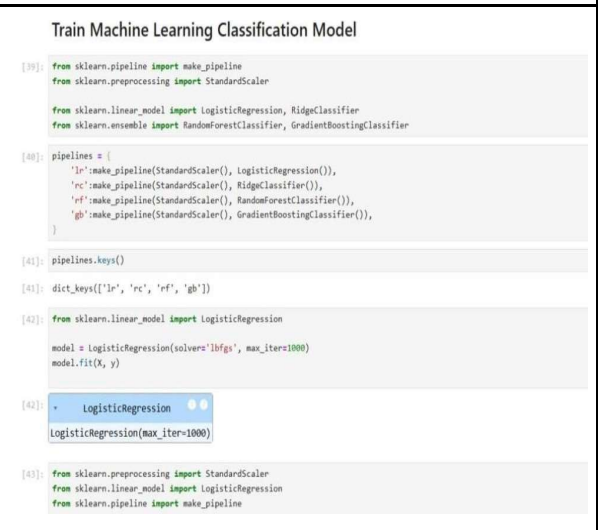
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

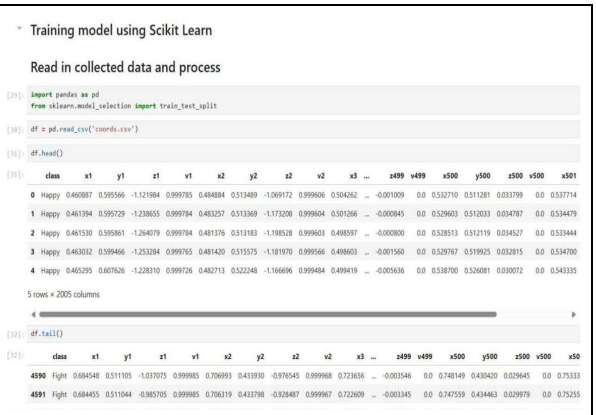
Initial Model Training Code (5 marks):

Paste the screenshot of the model traning code

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

Model 1	Gradient Boosting is the grouping of Gradient descent and Boosting. In gradient boosting, each new model minimizes the loss function from its predecessor using the Gradient Descent Method. This procedure continues until a more optimal estimate of the target variable has been achieved	
Model 2	Logistic Regression is used for binary classification tasks, predicting outcomes like spam vs. non-spam or disease vs. no disease, and it provides probabilities of class membership. It also helps in understanding feature importance and serves as a strong baseline model due to its simplicity and interpretability.	
Model 3	The Random Forest Classifier is ideal because it combines the strength of multiple decision trees, offering high accuracy, robust performance against overfitting, and the ability to handle large and complex datasets, ensuring reliable predictions in varied shipping scenarios.	

Model 4	The file "coords.csv" into a Pandas DataFrame. It then splits the data into training and testing sets using the train_test_split function from Scikit-learn. The df.head() and df.tail() functions are used to display the first and last few rows of the DataFrame, respectively.	

Training model using Scikit Learn

Read in collected data and process

```
[29]: import pandas as pd
      from sklearn.model_selection import train_test_split

[30]: df = pd.read_csv('coords.csv')

[31]: df.head()
```

[31]:

	class	x1	y1	z1	v1	x2	y2	z2	v2	x3	...	z499	v499	x500	y500	z500	v500	x501
0	Happy	0.460887	0.595566	-1.121984	0.999785	0.484884	0.513489	-1.069172	0.999606	0.504262	...	-0.001009	0.0	0.532710	0.511281	0.033799	0.0	0.537714
1	Happy	0.461394	0.595729	-1.238655	0.999784	0.483257	0.513369	-1.173208	0.999604	0.501266	...	-0.000845	0.0	0.529603	0.512033	0.034787	0.0	0.534479
2	Happy	0.461530	0.595861	-1.264079	0.999784	0.481376	0.513183	-1.198528	0.999603	0.498597	...	-0.000800	0.0	0.528513	0.512119	0.034527	0.0	0.533444
3	Happy	0.463032	0.599466	-1.253284	0.999765	0.481420	0.515575	-1.181970	0.999566	0.498603	...	-0.001560	0.0	0.529767	0.513925	0.032815	0.0	0.534700
4	Happy	0.465295	0.607626	-1.228310	0.999726	0.482713	0.522248	-1.166696	0.999484	0.499419	...	-0.005636	0.0	0.538700	0.526081	0.030072	0.0	0.543335

5 rows × 2005 columns

```
[32]: df.tail()
```

[32]:

	class	x1	y1	z1	v1	x2	y2	z2	v2	x3	...	z499	v499	x500	y500	z500	v500	x50
4590	Fight	0.684548	0.511105	-1.037075	0.999985	0.706993	0.433930	-0.976545	0.999968	0.723656	...	-0.003546	0.0	0.748149	0.430420	0.029645	0.0	0.75333
4591	Fight	0.684455	0.511044	-0.985705	0.999985	0.706319	0.433798	-0.928487	0.999967	0.722609	...	-0.003345	0.0	0.747559	0.434463	0.029979	0.0	0.75255