Q1. Describe how you built your model, how you used the training set, how you choose hyperparameters and why you made the choices you did.

Data pre-processing

We manually sanitised the dataset.

1. The first few columns of the dataset (`user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`), which are not relevant to the problem, are removed.
2. `yes`/`no` in the `new_window` column are substituted with with `1`/`0`.
3. Columns with empty values and errors are dropped.
4. Rows containing erroneous data in the `class` column are dropped. `A`/`B`/`C`/`D`/`E` are then substituted with `1`/`2`/`3`/`4`/`5`.
5. Erroneous data after the `class` column are dropped.

After pre-processing and sanitisation, a `19622×55` CSV matrix (excluding labels) is obtained.

Model

We employed a random forest classifier.

> The two hyperparameters we tuned are `n_estimators` and `test_size`.
>
> ```
> # Hyperparameters
> N_ESTIMATORS = 200
> TEST_SIZE = 0.2
> ```

For `n_estimators`, we incremented it by `50` beginning from `100`. We used `n_estimators = 200` in the end because it can consistently achieve >95% accuracy in testing.

For `test_size`, the model is trained on 80% of the training dataset with `test_size = 0.2` while the rest is split to test the model.

The hyperparameters we tuned already yielded a very high accuracy, so no further optimisations are employed. Default hyperparameters settings are used for the rest.

Q2. What are the important features of the training set? How you obtain the results?

Feature importance

Feature importance is acquired with `model.feature_importances_`.

Feature label	Importance
num_window	0.092442
roll_belt	0.074710
yaw_belt	0.051930
magnet_dumbbell_z	0.045757
pitch_forearm	0.045721
magnet_dumbbell_y	0.043380
pitch_belt	0.041529
roll_forearm	0.035012
magnet_dumbbell_x	0.030000
roll_dumbbell	0.026281
magnet_belt_y	0.025142
accel_dumbbell_y	0.024722
accel_belt_z	0.024186
magnet_belt_z	0.022496
accel_dumbbell_z	0.020494
accel_forearm_x	0.019814
roll_arm	0.017512
magnet_belt_x	0.016070
yaw_dumbbell	0.015871
magnet_arm_x	0.015695
total_accel_dumbbell	0.015565
gyros_belt_z	0.015538
magnet_forearm_z	0.015358
accel_dumbbell_x	0.014625

Feature label	Importance
gyros_dumbbell_y	0.014411
accel_forearm_z	0.014018
accel_arm_x	0.013997
total_accel_belt	0.013754
yaw_arm	0.012924
magnet_arm_y	0.012513
magnet_forearm_x	0.011880
magnet_forearm_y	0.011529
pitch_dumbbell	0.010522
pitch_arm	0.010240
yaw_forearm	0.009970
magnet_arm_z	0.009346
accel_arm_y	0.008292
accel_belt_y	0.008053
accel_forearm_y	0.007447
gyros_arm_y	0.007316
gyros_arm_x	0.007274
gyros_belt_y	0.007097
accel_belt_x	0.006923
gyros_dumbbell_x	0.006735
accel_arm_z	0.006690
gyros_forearm_y	0.006485
total_accel_arm	0.005695
total_accel_forearm	0.005571
gyros_belt_x	0.005471
gyros_dumbbell_z	0.004686
gyros_forearm_z	0.004218
gyros_forearm_x	0.003909
gyros_arm_z	0.003128
new_window	0.000053

> The code responsible is as follows.

```
debugLog("main", "printing feature importance")
feature_importances = pd.DataFrame(
    model.feature_importances_, index=training_x.columns, columns=["importance"]
).sort_values("importance", ascending=False)
print(feature_importances)
```

Model metrics

Metric	Value
Accuracy	0.9982165605095541
Precision	0.9982171198022268
Recall	0.9982165605095541
Confusion matrix	[1097 0 0 0 0] [2 731 0 0 0] [0 1 685 0 0] [0 0 2 668 1] [0 0 0 1 737]

The code responsible is as follows.

```
debugLog("main", "printing testing metrics")
testing_prediction_y = model.predict(testing_x)
print("Accuracy:", metrics.accuracy_score(testing_y, testing_prediction_y))
print(
    "Precision:",
    metrics.precision_score(
        testing_y, testing_prediction_y, average="weighted", zero_division=0
    ),
)
print(
    "Recall:",
    metrics.recall_score(
        testing_y, testing_prediction_y, average="weighted", zero_division=0
    ),
)
print("Confusion Matrix:")
print(
    metrics.confusion_matrix(
        testing_y, testing_prediction_y, labels=[1, 2, 3, 4, 5]
    )
)
```

Prediction

The predicting dataset is pre-processed and sanitised in the same manner as the training dataset. The model predicted the classes of the 20 unseen data entries as `[A B D B B E D E A B A A C A D B C E D B]`.

> The code responsible is as follows.

```
debugLog("main", "predicting")
predictingprediction_y = model.predict(predicting_x)
#
debugLog("main", "printing prediction results")
print(predictingprediction_y)
# replace 1 2 3 4 5 with A B C D E
print(
    pd.Series(predictingprediction_y)
    .replace([1, 2, 3, 4, 5], ["A", "B", "C", "D", "E"])
    .tolist()
)
```

Terminal output

Some lines are omitted for brevity, indicated by ellipses.

```
20-11-2023 04:45:47 | prepare ⇒ trial hash: 0×2e53cdcd3e5aa71c
20-11-2023 04:45:47 | prepare ⇒ output path: ./output/20112023_044547_0×2e53cdcd3e5aa71c/
20-11-2023 04:45:47 | main ⇒ starting
20-11-2023 04:45:47 | main ⇒ loading dataset
20-11-2023 04:45:47 | main ⇒ building model
20-11-2023 04:45:47 | main ⇒ training model
building tree 1 of 200
building tree 2 of 200
building tree 3 of 200
building tree 4 of 200
building tree 5 of 200
 ...
building tree 196 of 200
building tree 197 of 200
building tree 198 of 200
building tree 199 of 200
building tree 200 of 200
20-11-2023 04:46:12 | main ⇒ printing feature importance
                      importance
num_window              0.092442
roll_belt               0.074710
yaw_belt                0.051930
magnet_dumbbell_z       0.045757
pitch_forearm           0.045721
 ...
gyros_dumbbell_z        0.004686
gyros_forearm_z         0.004218
gyros_forearm_x         0.003909
gyros_arm_z             0.003128
new_window              0.000053
20-11-2023 04:46:12 | main ⇒ printing training metrics
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
Confusion Matrix:
[[4483    0    0    0    0]
 [   0 3063    0    0    0]
 [   0    0 2737    0    0]
 [   0    0    0 2545    0]
 [   0    0    0    0 2869]]
20-11-2023 04:46:13 | main ⇒ printing testing metrics
Accuracy: 0.9982165605095541
Precision: 0.9982171198022268
Recall: 0.9982165605095541
Confusion Matrix:
[[1097    0    0    0    0]
 [   2  731    0    0    0]
 [   0    1  685    0    0]
 [   0    0    2  668    1]
 [   0    0    0    1  737]]
20-11-2023 04:46:13 | main ⇒ predicting
20-11-2023 04:46:13 | main ⇒ printing prediction results
[1 2 4 2 2 5 4 5 1 2 1 1 3 1 4 2 3 5 4 2]
['A', 'B', 'D', 'B', 'B', 'E', 'D', 'E', 'A', 'B', 'A', 'A', 'C', 'A', 'D', 'B', 'C', 'E', 'D', 'B']
20-11-2023 04:46:13 | main ⇒ finished
```