Bhargavi Poyekar
CH33454

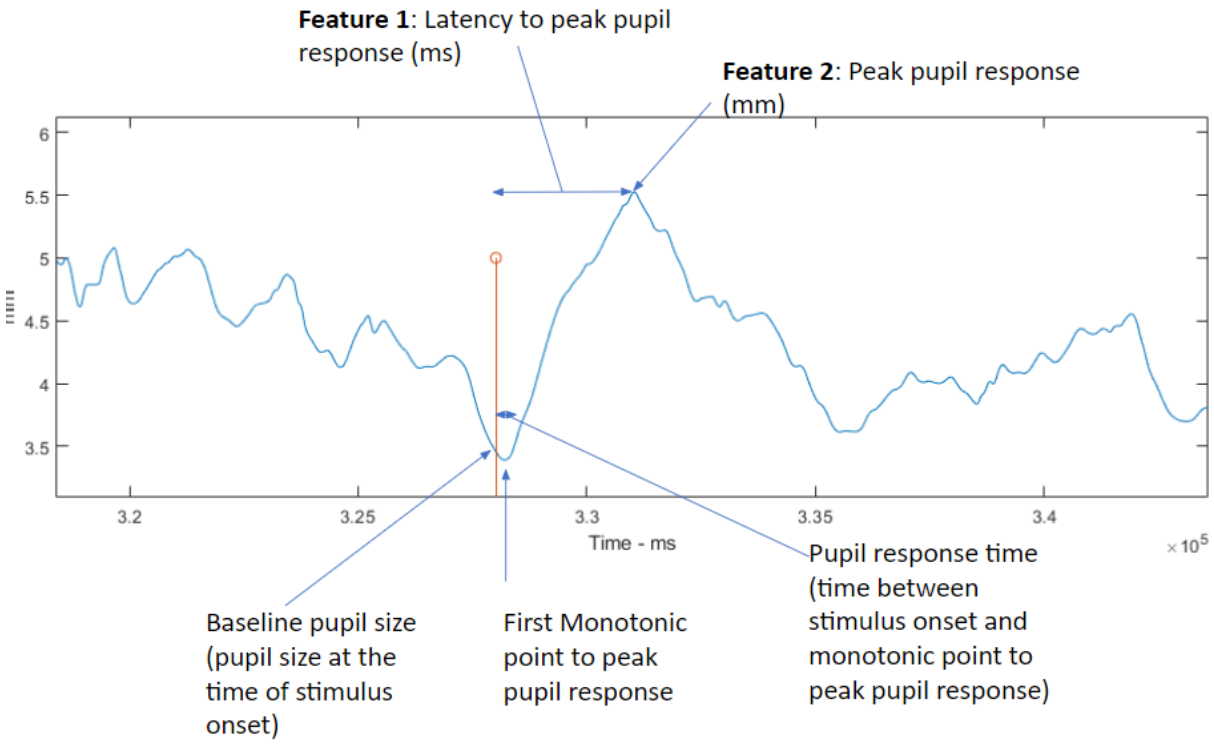
CompPsych Project 2
Final Report

Step 3: Extract Features and Modelling

a. Extract Features

For every trial of each subject, I extracted 4 features: peak pupil response, latency to peak, velocity, and acceleration.

I used the following reference, given by the professor to extract the first 2 features.



- The **latency to the peak** is the time taken in ms to reach the peak pupil response, after the stimulus onset i.e the start of the trial.
- The **peak pupil response** (in mm) is the maximum diameter the pupil expands to after the start of the trial till the response is given.

I calculated both these features using the trial vs time table I formed in step 2:

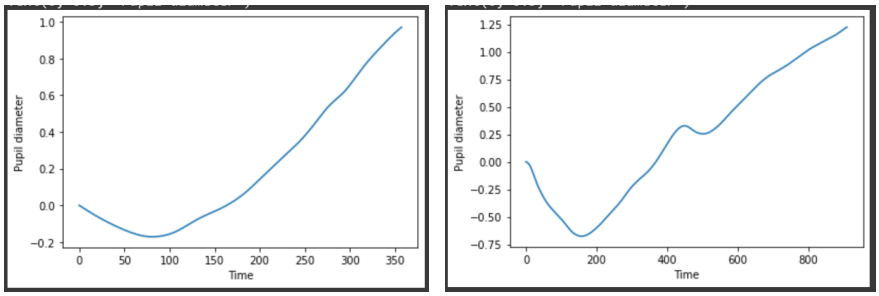
	0	1	2	3	4	5	6	7	8	9	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
0	0.0	-0.001284	-0.002765	-0.004494	-0.006521	-0.008919	-0.011712	-0.014920	-0.018600	-0.022778	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	0.0	-0.000251	-0.000421	-0.000508	-0.000515	-0.000443	-0.000288	-0.000048	0.000277	0.000691	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	0.0	0.001053	0.002226	0.003520	0.004930	0.006457	0.008097	0.009850	0.011713	0.013683	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	0.0	-0.000066	-0.000129	-0.000187	-0.000240	-0.000286	-0.000326	-0.000360	-0.000388	-0.000412	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.0	-0.002823	-0.005647	-0.008465	-0.011272	-0.014059	-0.016821	-0.019547	-0.022228	-0.024856	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	0.0	-0.005240	-0.010607	-0.015663	-0.019912	-0.022858	-0.024459	-0.025191	-0.025776	-0.026598	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1996	0.0	-0.000605	-0.001928	-0.004684	-0.009686	-0.017414	-0.027643	-0.039411	-0.051315	-0.061972	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1997	0.0	-0.006970	-0.018514	-0.033416	-0.049465	-0.064361	-0.076084	-0.083530	-0.087272	-0.089917	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1998	0.0	0.002532	0.004907	0.006660	0.007093	0.005542	0.001775	-0.003865	-0.010717	-0.018175	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1999	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2000 rows × 2000 columns

In the table above, the rows represent the trials for each subject (40 each for 50 subjects), and the columns represent the relative time after the stimulus onset. The cell values represent the pupil diameter in mm.

I calculated the peak pupil response by finding the maximum value for each row.

Professor suggested the scipy signal's find_peak method, but when I tried using it, for many trials, no peaks were detected. So, I plotted the diameters.



(a) No peak found (b) Peak found

Many trials have diameters as shown in the above figure (a), which don't have any mountain-type peak, and hence didn't return any peaks for the find_peak function. Whereas, the trials, where the peaks were returned, had diameters similar to the figure shown in (b), which detected the peak, found at around 400 ms, which is not the one that we need.

So, I decided to directly take the max of all the trials to get the peak pupil response, and the index*4 of the column to get the latency to peak, as the time interval for each pupil diameter was 4 ms.

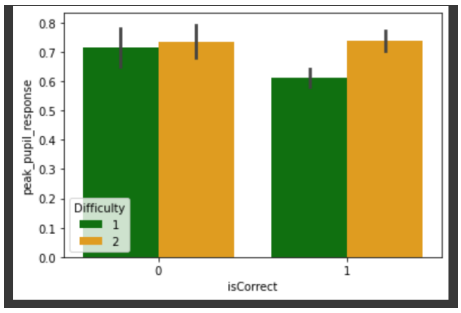
- **Velocity** (mm/ms): I calculated this using the first 2 features, to get the change in pupil diameter per unit time. (Peak pupil response/Latency to peak)
- **Acceleration** mm/ms^2: Velocity/ Latency to the peak.

Extracted Feature Table:

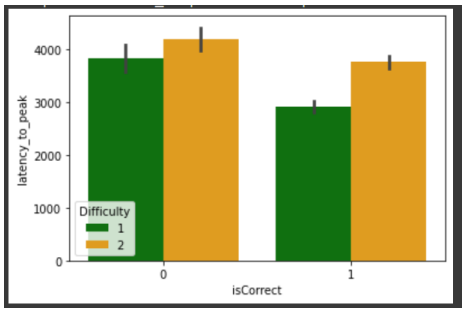
	subject_id	trial_no	Stimulus_onset	RT	Difficulty	isCorrect	peak_pupil_response	latency_to_peak	velocity	acceleration
0	subject_01_pupil	1	1598.902	3635.104895	1	1	1.224777	3632	0.000337	9.284646e-08
1	subject_01_pupil	2	23814.206	2853.791475	2	1	1.806894	2848	0.000634	2.227679e-07
2	subject_01_pupil	3	39763.242	2182.772636	2	1	0.685657	1548	0.000443	2.861311e-07
3	subject_01_pupil	4	61978.531	3180.494785	2	0	0.734249	3176	0.000231	7.279179e-08
4	subject_01_pupil	5	79677.448	3880.589724	2	1	1.378393	3876	0.000356	9.174988e-08
...	...	...	...	...	...	...	...	...	...	...
1995	subject_52_pupil	36	657183.662	3118.596554	1	1	0.188448	2356	0.000080	3.395009e-08
1996	subject_52_pupil	37	673532.666	1828.583717	1	1	0.014736	1824	0.000008	4.429247e-09
1997	subject_52_pupil	38	693064.791	4305.474758	1	1	1.822619	4168	0.000437	1.049157e-07
1998	subject_52_pupil	39	711630.330	4800.941467	2	1	1.890446	3476	0.000544	1.564605e-07
1999	subject_52_pupil	40	726896.064	2943.220139	1	1	1.166643	2812	0.000415	1.475392e-07

2000 rows × 10 columns

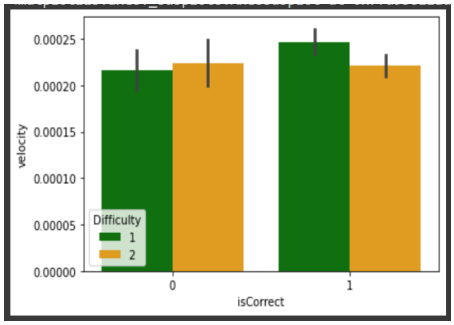
Visualization of features:



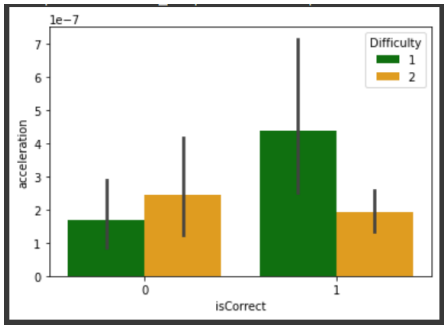
(a) Peak pupil response



(b) latency to peak



(c) Velocity



(d) acceleration.

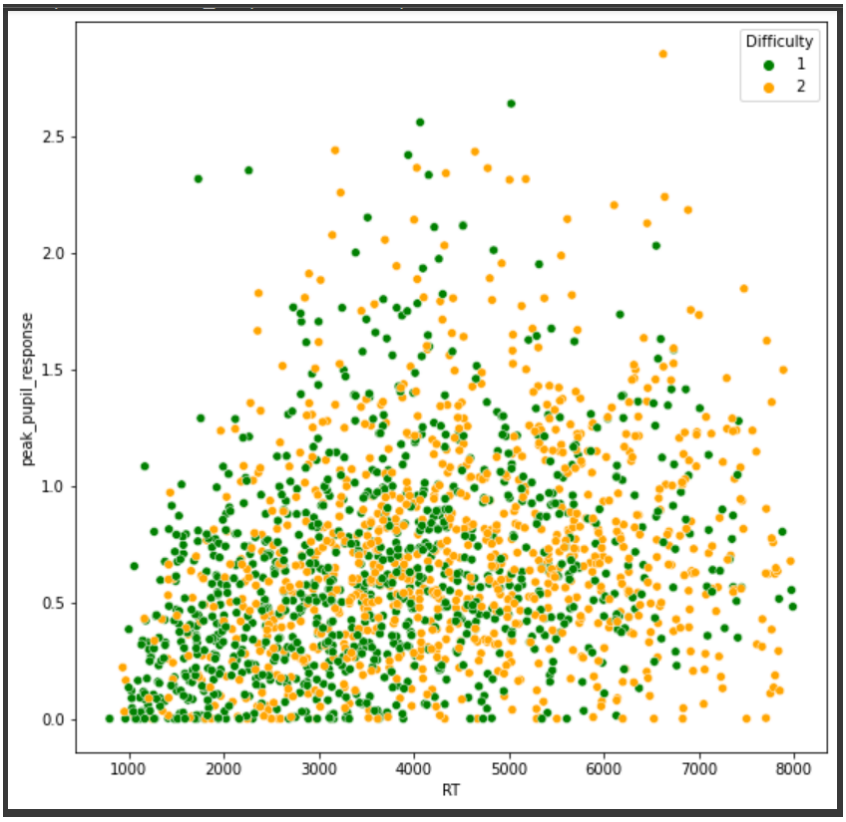
The above 4 figures show the features vs isCorrect and keeping the hue as Difficulty to understand the relationship between these 3 attributed.

From fig (a), we can see that the average peak pupil response is lower for difficulty 1 as compared to difficulty 2. While avg peak pupil response is slightly lower for wrong response compared to correct response.

From fig (b), we can see that the avg latency to the peak is lower for difficulty 1 as compared to difficulty 2. While avg latency to the peak is slightly higher for wrong response compared to correct response.

From fig (c), we can see that the velocity is lower for difficulty 1 when the answer is wrong and higher when the answer is correct. Avg velocity is almost similar for both correct and wrong responses.

From fig (d), we can see that the acceleration is lower for difficulty 1 when the answer is wrong and higher when the answer is correct. Avg acceleration is lower for wrong responses.



The figure above shows, response time vs peak pupil response. There is no clear trend, but it looks like the peak pupil response is more for higher response time. Also, most of the difficulty level 1 has lower RT than compared to difficulty level 2.

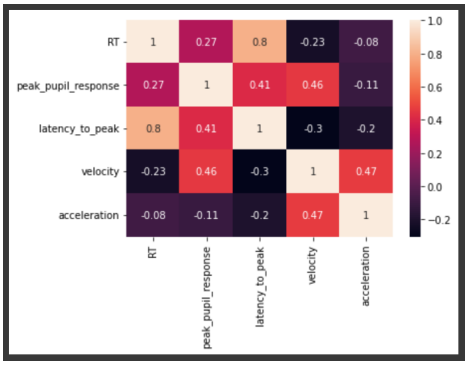
	peak_pupil_response	latency_to_peak	velocity	acceleration
count	2000.000000	2000.000000	2000.000000	2.000000e+03
mean	0.684672	3486.178000	0.000225	2.867978e-07
std	0.466999	1790.646455	0.000184	2.185240e-06
min	0.000000	0.000000	0.000000	0.000000e+00
25%	0.343013	2267.000000	0.000113	2.569957e-08
50%	0.618219	3460.000000	0.000190	5.478414e-08
75%	0.941370	4709.000000	0.000289	1.074813e-07
max	2.852988	7996.000000	0.002490	7.781055e-05

The figure above shows the statistics of the features.

b. Modeling for Response Time.
https://colab.research.google.com/drive/1bEsLkkotZ_QLnEGaWMU9VmTFFn6oQLG-?usp=sharing

The data was split into train and validation 1600 in the train set and 400 in the validation set, and then the data was normalized using standard Scalar.

I used Regression models for the prediction of Response Times. I tried different regression models like Linear Regression, Random Forest Regressor, Decision Tree Regressor, SVM Regressor, SGD Regressor. Regularization (Lasso and Ridge) was also used, but they gave the same results as Linear Regression.



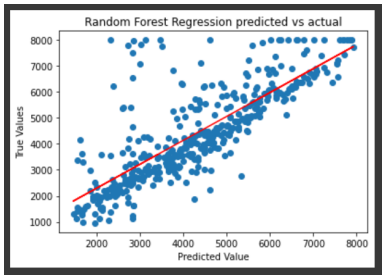
The heatmap shows the correlation between the independent features and the dependent feature RT. It shows that latency_to_peak is most correlated with RT, which I think is because most of the time, the pupil response is maximum near the response time.

I also used PCA, and hyper tuned the model to get better results.
I used 3 metrics for the comparison of models. Mean Squared Error (mse), Mean Absolute Error (mae), and Root Mean Squared Error (rmse)

Errors on Validation data:

	Linear Regression	**Random Forest**	**Decision Tree**	**SVM**	**SGD**
Normal	mae: 810.03	mae: 960.32	mae: 911.07	mae:1029.36	mae: 821.26
	rmse:1209.013	rmse:1320.72	rmse:1471.03	rmse:1369.32	rmse:1224.47
With PCA	mae: 841.90	Mae: 1495.35	mae: 825.19	mae: 1040.24	mae: 854.98
	rmse:1234.04	rmse:1156.77	rmse:1310.87	rmse:1380.58	rmse:1240.06
Hyperparameter		mae: 724.49		mae: 777.317	mae: 877.1563

tuning		rmse:1134.86		Rmse: 1357.05	Rmse: 1243.78

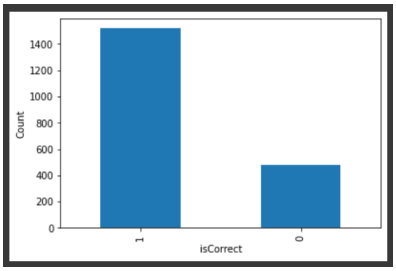


This is the prediction vs actual plot for the winning model: Hyperparameter Tuned Random Forest Regressor (n_estimators: 150) with MAE of 724.49 and RMSE of 1134.86.

I have also plotted prediction vs actual for all the models, but not added in the report.

c. Modeling for isCorrect:
https://colab.research.google.com/drive/1oZieGIe7ydJQFlzIjwPy5m5gLDHYy437?usp=sharing

For isCorrect feature, the data is highly imbalanced. The samples for a correct answer are about 3 times in count than the wrong answers.



So, I used random oversampling to create more data for the wrong answer. This increased the total samples from 2000 to 3036 and then divided the data into train validation split of 80:20.

I have then used many classification models like, Logistic Regression, Random Forest, KNN, Gaussian NB, Decision Tree, and Adaboost.

I used various comparison metrics like, accuracy, roc_auc scores, precision, recall, and confusion matrix.

Accuracy and Roc Auc Scores:

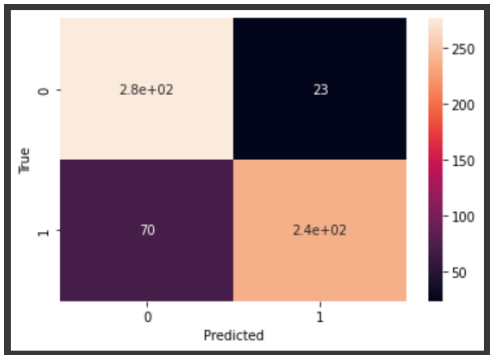
	Logistic Regression	Random Forest	KNN	Gaussian NB	Decision Tree	Adaboost
Normal	Accuracy:0.5131	0.8273	0.6677	0.4901	0.8009	0.5838
	Roc-AUC: 0.5693	0.9188	0.7460	0.5643	0.8155	0.6295
With PCA	0.5164	0.8470	0.6776	0.5098	0.8141	0.5674
	0.5633	0.9261	0.7469	0.5127	0.8290	0.6005
Hyperparameter tuning	0.5131	0.8289	0.7121	0.4917		0.6430
	0.5694	0.9186	0.7823	0.5641		0.6998

Random forest with PCA was the winning model with an accuracy of 0.8470 and ROC AUC score of 0.9261.

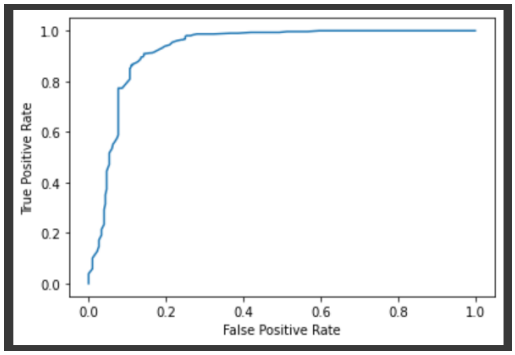
Classification Report:

```
              precision    recall  f1-score   support

           0       0.80      0.92      0.86       300
           1       0.91      0.77      0.84       308

    accuracy                           0.85       608
   macro avg       0.86      0.85      0.85       608
weighted avg       0.86      0.85      0.85       608

Validation Accuracy: 0.8470394736842105
ROC_AUC Score:0.9261688311688311
[[277  23]
 [ 70 238]]
```

Confusion Matrix:



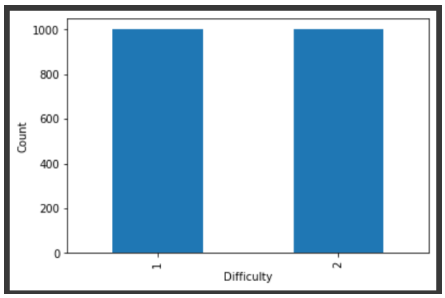
ROC Curve:



d. Modeling for Difficult:

https://colab.research.google.com/drive/1eUULpRrL_YMjW_t_X2y-4N1qRcchxxcQ?usp=sharing

The data is balanced for Difficult feature.



So, I applied the classification models to the data, the metrics were the same as isCorrect features.

Accuracy and Roc Auc Scores:

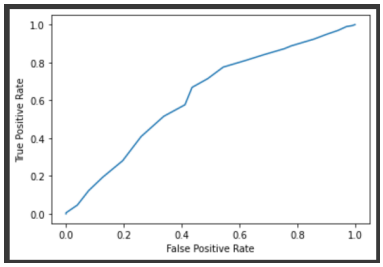
	Logistic Regression	**Random Forest**	**KNN**	**Gaussian NB**	**Decision Tree**	**Adaboost**
Normal	Accuracy: 0.575	0.51	0.545	0.5025	0.495	0.5875
	Roc-AUC: 0.6103	0.5233	0.5501	0.6084	0.4960	0.5961
With PCA	0.58	0.5525	0.55	0.525	0.5275	0.565
	0.6101	0.5652	0.5565	0.5615	0.5298	0.5914
Hyperparameter tuning	0.575	0.5775	0.615	0.5		0.58
	0.6103	0.6084	0.6226	0.6085		0.6091

KNN + Hyperparameter Tuning was the best model, with the best parameter of n_neighbors: **35**, **Accuracy of 0.615** and **Roc Auc of 0.6226.**

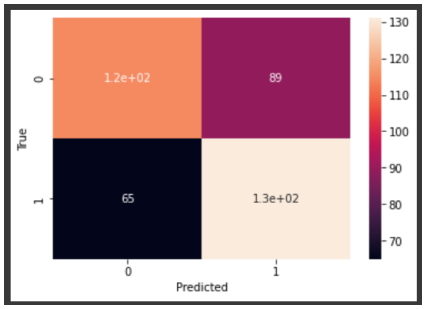
Classification Report:

```
              precision    recall  f1-score   support

           1       0.64      0.56      0.60       204
           2       0.60      0.67      0.63       196

    accuracy                           0.61       400
   macro avg       0.62      0.62      0.61       400
weighted avg       0.62      0.61      0.61       400

Validation Accuracy: 0.615
ROC_AUC Score:0.622624049619848
[[115  89]
 [ 65 131]]
```

Roc-Curve:



Confusion Matrix:



Step 4: Prediction on Test Data

I extracted features from test data, and made a similar table. The test data had 40 trials. This data is then used for the final prediction. I have used all the models to predict the data but adding the predictions of only the winning model here.

A. RT prediction: For reaction time metric, the mean absolute difference of the test RTs and actual RTs came around 1073.6ms and the r2 value is 0.472.

```
[3132.92308864, 4839.26676583, 4001.08084537, 6469.36316657,
       4335.32930365, 3354.06183721, 3588.28464019, 4554.85926455,
       3482.60642369, 3230.4543827 , 2282.63172786, 1691.68892547,
       2792.04284011, 3719.22677683, 3219.82458123, 3639.70744605,
       1706.52450883, 3029.77366295, 3098.38113305, 3720.42769898,
       1500.99656902, 2173.52150139, 4528.35504208, 2702.65726902,
       4229.62845008, 6431.9904406 , 3145.99361738, 5542.48313917,
       1690.85871542, 1514.39677563, 1506.70190627, 2541.50191644,
       5258.43518735, 1917.83397047, 3372.00280813, 3256.57291239,
       4408.83183327, 5598.15236886, 6390.08245463, 2076.93544375]
```

B. isCorrect Prediction: 28/40 = 70% - Accuracy

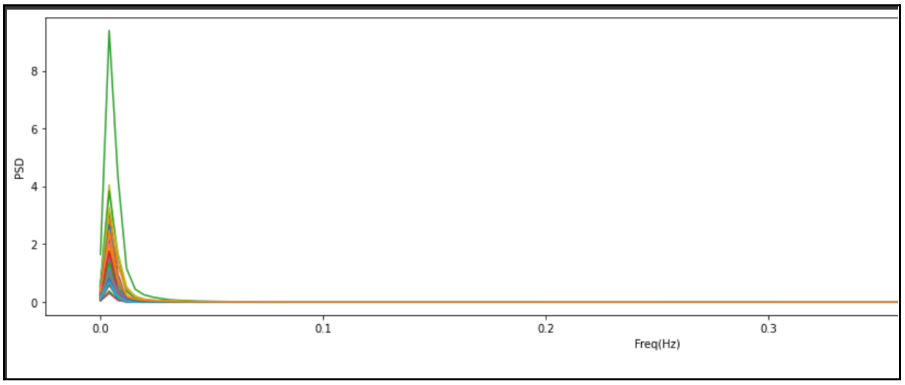
```
[1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

C. Difficulty prediction: 27/40 ~ 70%- Accuracy

```
[1 2 2 2 1 2 2 2 2 2 1 1 1 2 2 1 1 2 1 2 1 1 2 1 1 2 1 2 1 1 1 1 2 1
1 2 1 2 2 1]
```

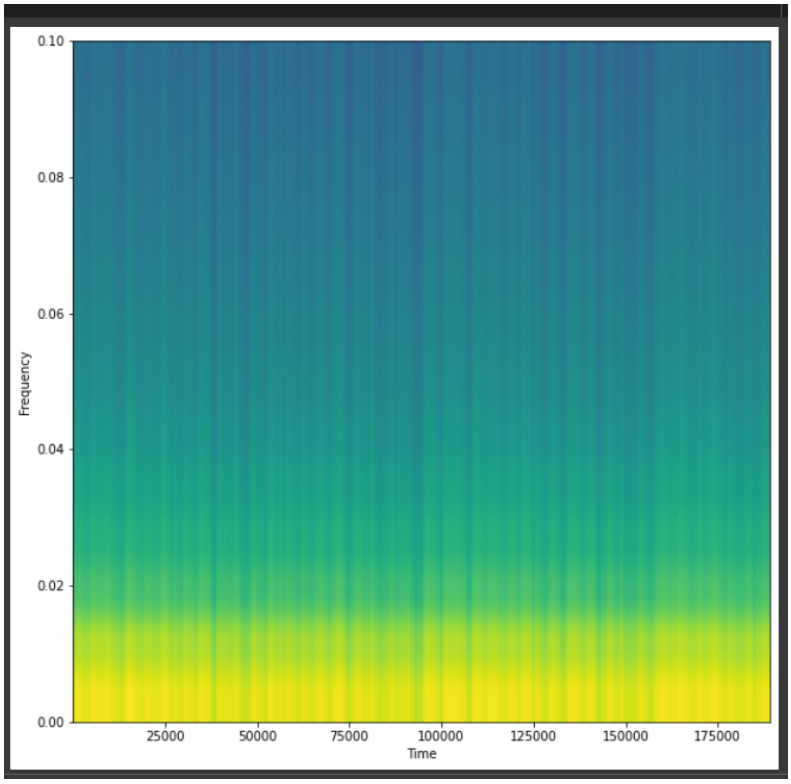
Step 5: Trying to predict Stimulus Onset Time:

I tried to look at the frequency domain PSD for all the subjects trial data.



The frequency with maximum PSD was around 0.2 Hz.

I also tried plotting the spectrogram.



This is the spectrogram for subject 1 and the spectrogram of other subjects was similar to this.

I tried plotting recurrence plots using pyRQA, pyts, pyunicorn, but these libraries gave me errors. I also tried to plot the recurrence plot by directly coding it using scipy distance metrics of pdist and squareform, but the data was too big, and my RAM limit was exceeded.