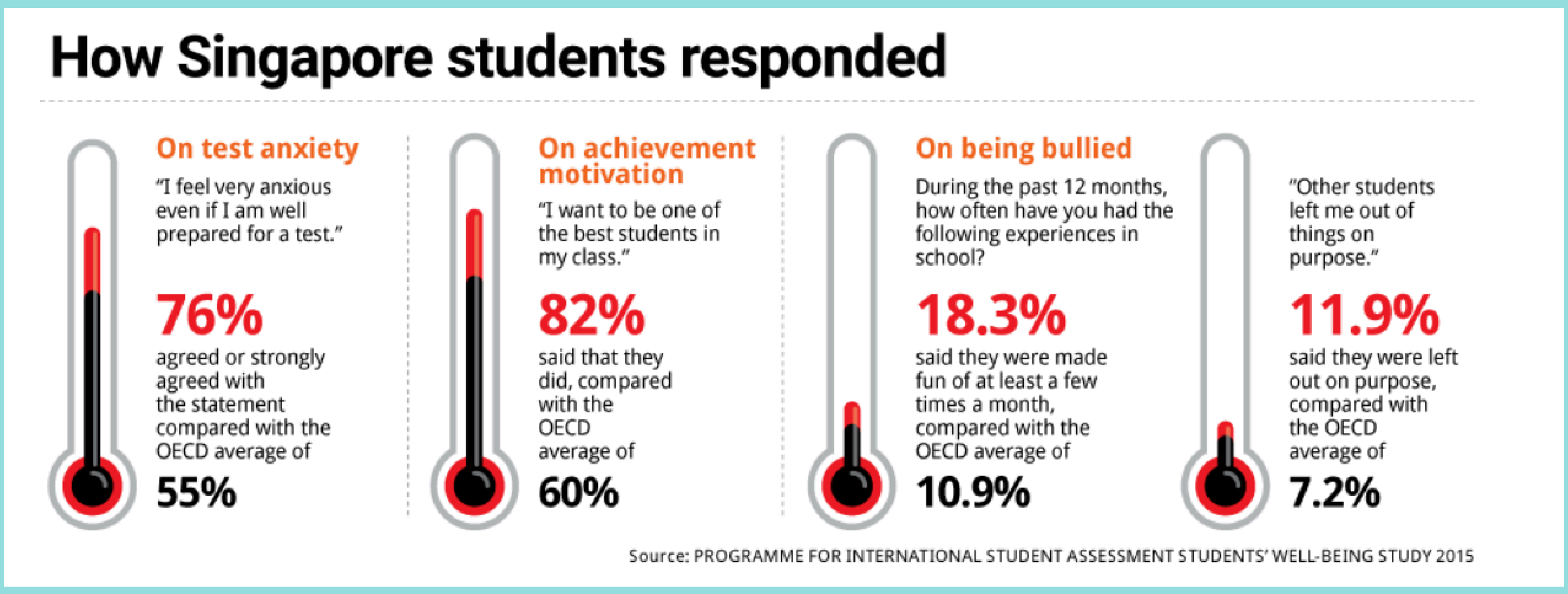
新民中學
XINMIN SECONDARY SCHOOL
堅韌不拔，服勤為衆. Steadfast in Spirit, Diligent in Service

Integrating AI to assess Students Well-being

Nguyen Le Hoang, Dinh Hoang Nam, Tan Yong Xiang

Problems

- Hard to recognise stress for the students
- Hard to evaluate students' stress level
- Many signs of stress are quantitative, it is harder to recognise the severity of stress



How Singapore students responded

On test anxiety

"I feel very anxious even if I am well prepared for a test."

76%

agreed or strongly agreed with the statement compared with the OECD average of

55%

On achievement motivation

"I want to be one of the best students in my class."

82%

said that they did, compared with the OECD average of

60%

On being bullied

During the past 12 months, how often have you had the following experiences in school?

18.3%

said they were made fun of at least a few times a month, compared with the OECD average of

10.9%

"Other students left me out of things on purpose."

11.9%

said they were left out on purpose, compared with the OECD average of

7.2%

Source: PROGRAMME FOR INTERNATIONAL STUDENT ASSESSMENT STUDENTS' WELL-BEING STUDY 2015

Solutions

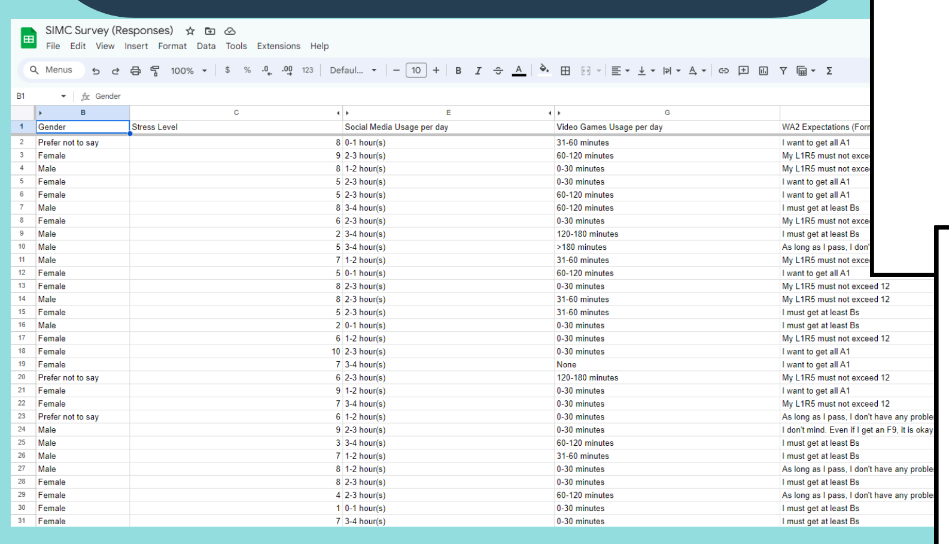
- Our aim: Using AI to identify student's well-being throughout their academic year based on their daily schedule
- Prompt: Build an AI model that can predict the student's stress level based on the 4 features we gathered: Sleeping time, Gaming time, Outdoor Activities time and Social Media Usage

Methodology

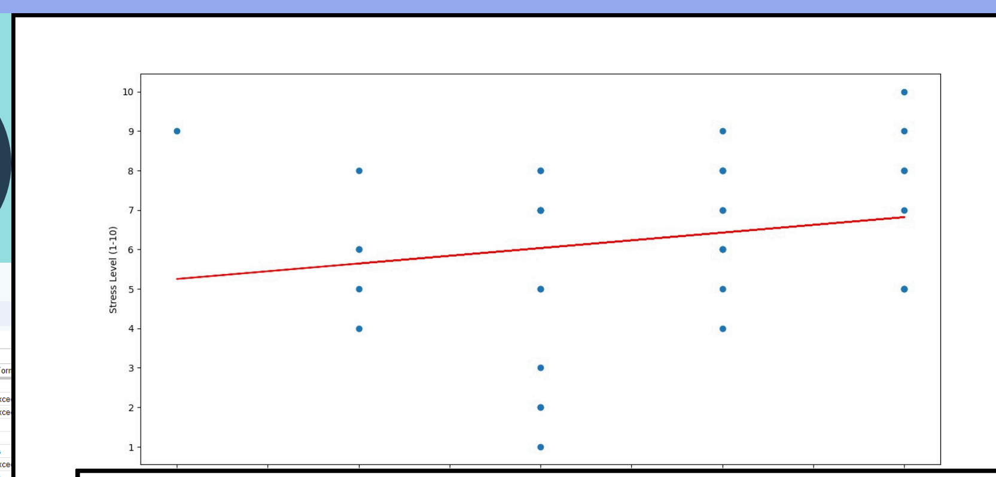
Overview:

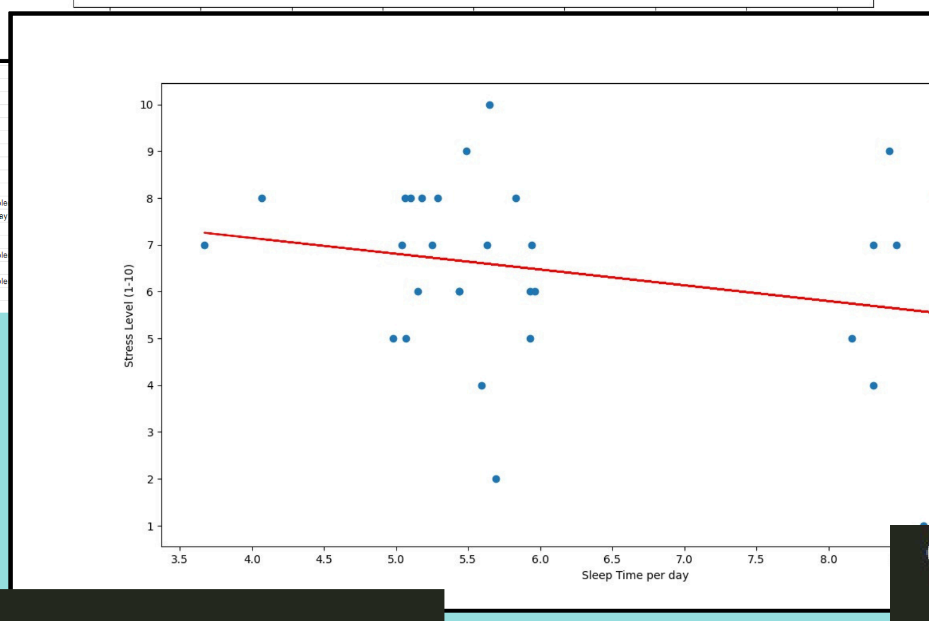
- Collect data from students
- Pre-processing data
- Split data
- Create a model to predict the stress level based on given features using Random Forest Classifier with GridSearchCV
- Train the model

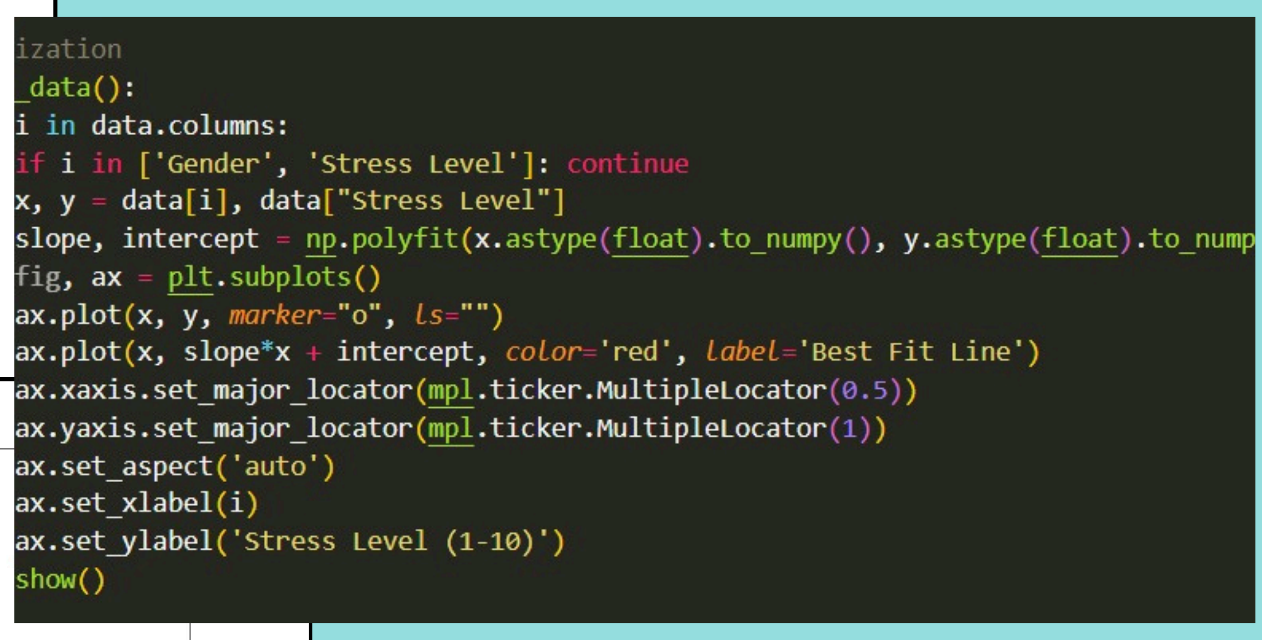
Data Collection



	B	C	E	G	
1	Gender	Stress Level	Social Media Usage per day	Video Games Usage per day	WA2 Expectations (Form
2	Prefer not to say	8	0-1 hour(s)	31-60 minutes	I want to get all A1
3	Female	9	2-3 hour(s)	60-120 minutes	My L1R5 must not exce
4	Male	8	1-2 hour(s)	0-30 minutes	My L1R5 must not exce
5	Female	5	2-3 hour(s)	0-30 minutes	I want to get all A1
6	Female	5	2-3 hour(s)	60-120 minutes	I want to get all A1
7	Male	8	3-4 hour(s)	60-120 minutes	I must get at least Bs
8	Female	6	2-3 hour(s)	0-30 minutes	My L1R5 must not exce
9	Male	2	3-4 hour(s)	120-180 minutes	I must get at least Bs
10	Male	5	3-4 hour(s)	>180 minutes	As long as I pass, I don'
11	Male	7	1-2 hour(s)	31-60 minutes	My L1R5 must not exce
12	Female	5	0-1 hour(s)	60-120 minutes	I want to get all A1
13	Female	8	2-3 hour(s)	0-30 minutes	My L1R5 must not exceed 12
14	Male	8	2-3 hour(s)	31-60 minutes	My L1R5 must not exceed 12
15	Female	5	2-3 hour(s)	31-60 minutes	I must get at least Bs
16	Male	2	0-1 hour(s)	0-30 minutes	I must get at least Bs
17	Female	6	1-2 hour(s)	0-30 minutes	My L1R5 must not exceed 12
18	Female	10	2-3 hour(s)	0-30 minutes	I want to get all A1
19	Female	7	3-4 hour(s)	None	I want to get all A1
20	Prefer not to say	6	2-3 hour(s)	120-180 minutes	My L1R5 must not exceed 12
21	Female	9	1-2 hour(s)	0-30 minutes	I want to get all A1
22	Female	7	3-4 hour(s)	0-30 minutes	My L1R5 must not exceed 12
23	Prefer not to say	6	1-2 hour(s)	0-30 minutes	As long as I pass, I don't have any proble
24	Male	9	2-3 hour(s)	0-30 minutes	I don't mind. Even if I get an F9, it is okay
25	Male	3	3-4 hour(s)	60-120 minutes	I must get at least Bs
26	Male	7	1-2 hour(s)	31-60 minutes	I must get at least Bs
27	Male	8	1-2 hour(s)	0-30 minutes	As long as I pass, I don't have any proble
28	Female	8	2-3 hour(s)	0-30 minutes	I must get at least Bs
29	Female	4	2-3 hour(s)	60-120 minutes	As long as I pass, I don't have any proble
30	Female	1	0-1 hour(s)	0-30 minutes	I must get at least Bs
31	Female	7	3-4 hour(s)	0-30 minutes	I must get at least Bs

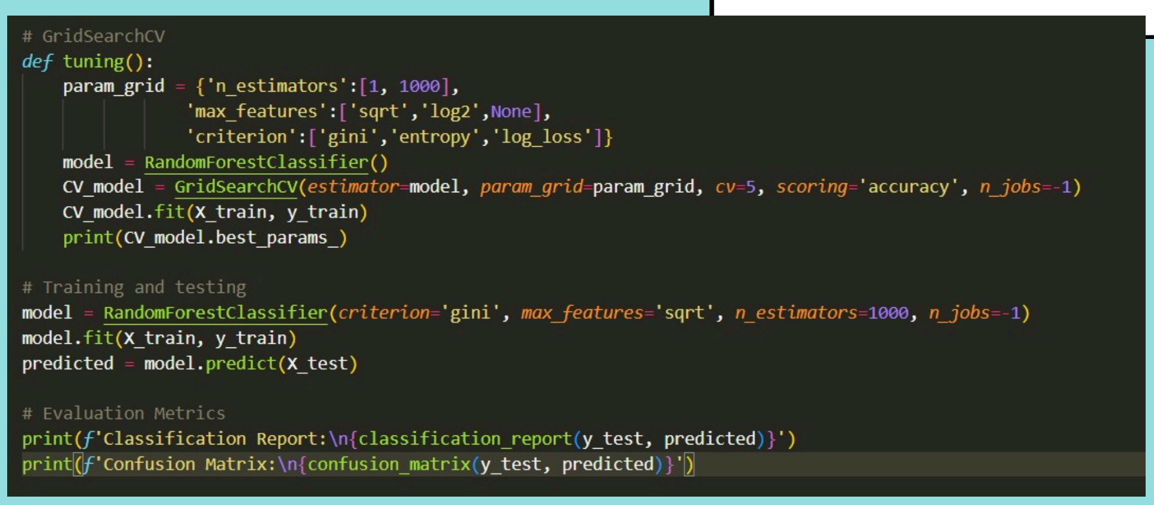






```
ization
_data():
i in data.columns:
if i in ['Gender', 'Stress Level']: continue
x, y = data[i], data["Stress Level"]
slope, intercept = np.polyfit(x.astype(float).to_numpy(), y.astype(float).to_nump
fig, ax = plt.subplots()
ax.plot(x, y, marker="o", ls="")
ax.plot(x, slope*x + intercept, color='red', label='Best Fit Line')
ax.xaxis.set_major_locator(mpl.ticker.MultipleLocator(0.5))
ax.yaxis.set_major_locator(mpl.ticker.MultipleLocator(1))
ax.set_aspect('auto')
ax.set_xlabel(i)
ax.set_ylabel('Stress Level (1-10)')
show()
```

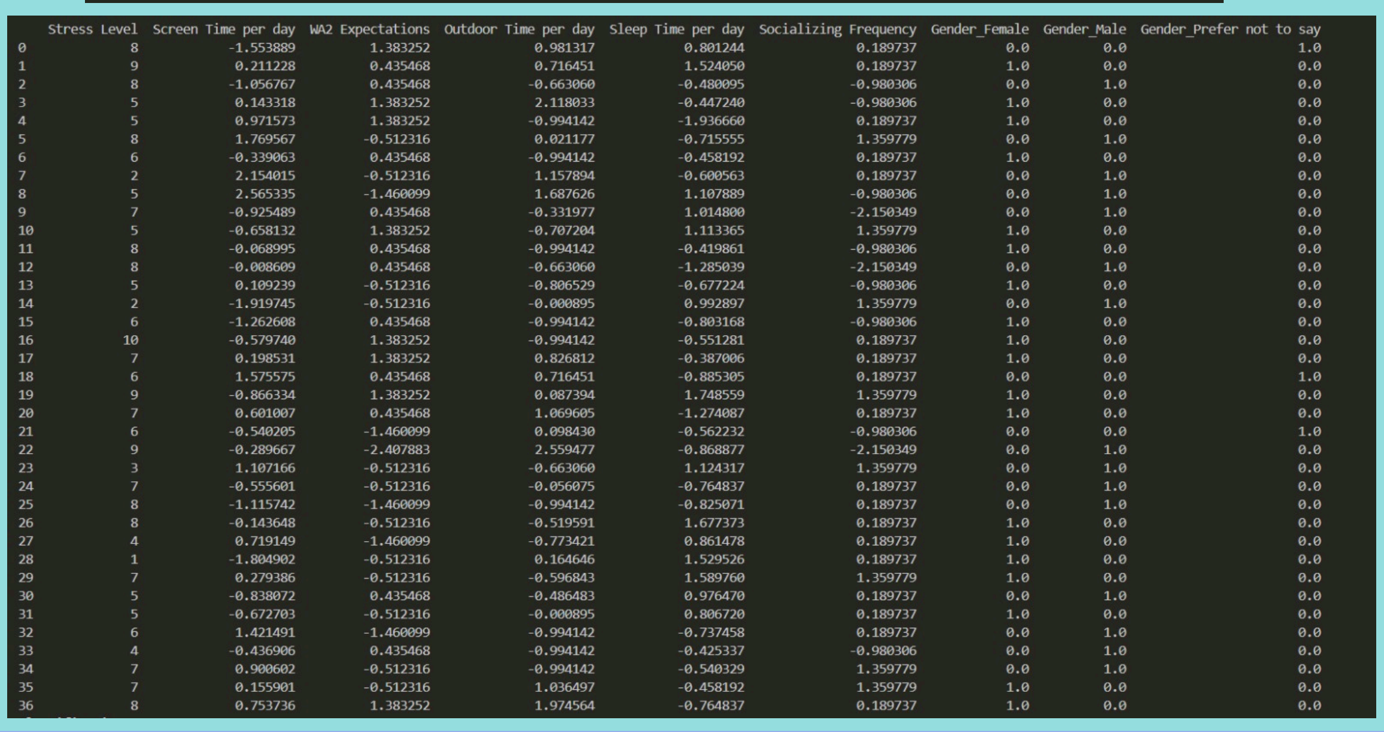
Model Training



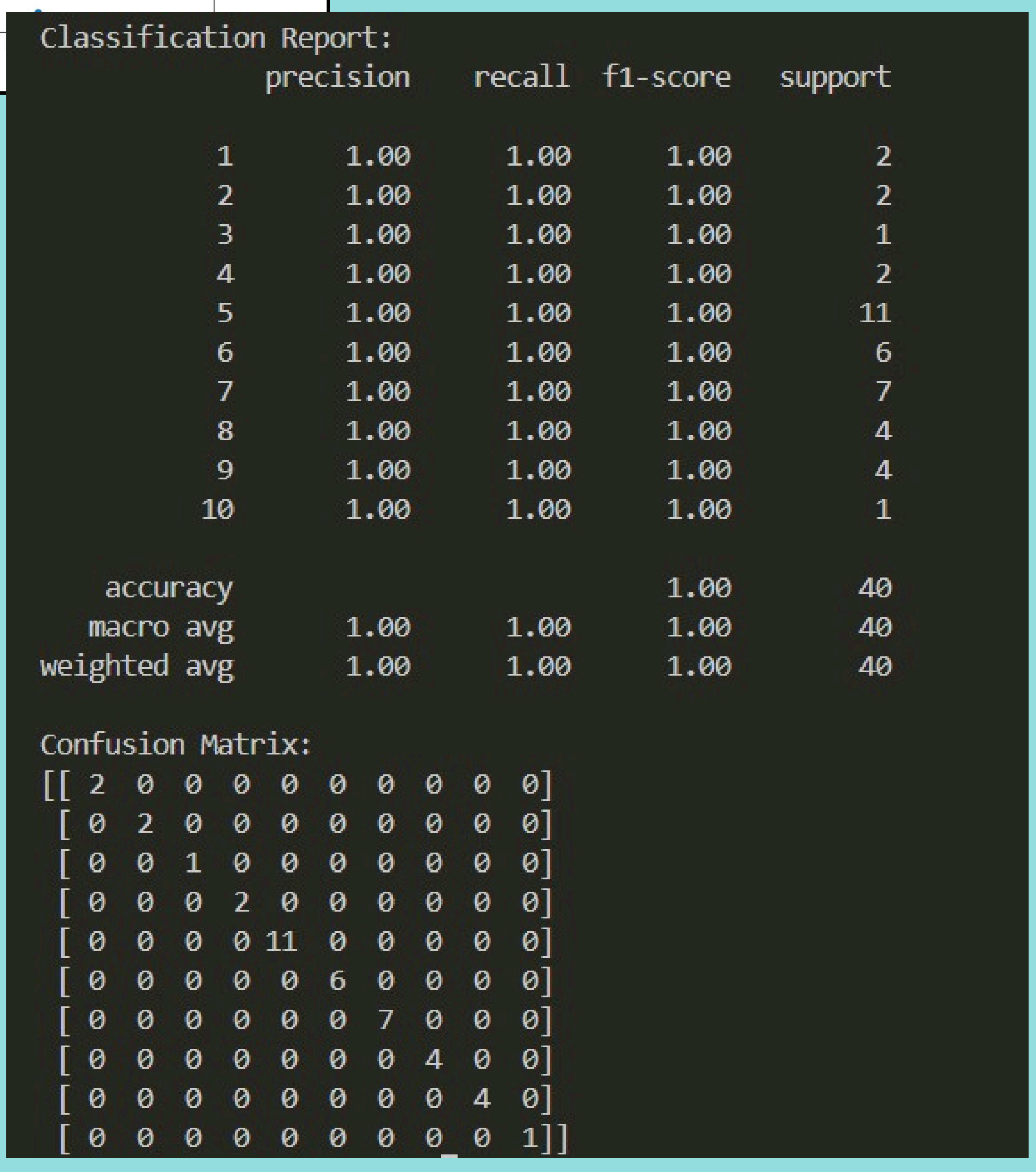
```
# GridSearchCV
def tuning():
    param_grid = {'n_estimators':[1, 1000],
                  'max_features':['sqrt','log2',None],
                  'criterion':['gini','entropy','log_loss']}
    model = RandomForestClassifier()
    CV_model = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
    CV_model.fit(X_train, y_train)
    print(CV_model.best_params_)

# Training and testing
model = RandomForestClassifier(criterion='gini', max_features='sqrt', n_estimators=1000, n_jobs=-1)
model.fit(X_train, y_train)
predicted = model.predict(X_test)

# Evaluation Metrics
print(f'Classification Report:\n{classification_report(y_test, predicted)}')
print(f'Confusion Matrix:\n{confusion_matrix(y_test, predicted)}')
```



```
    Stress Level  Screen Time per day  WA2 Expectations  Outdoor Time per day  Sleep Time per day  Socializing Frequency  Gender_Female  Gender_Male  Gender_Prefer not to say
0              8            -1.553889          1.383252              0.981317            0.801244               0.189737            0.0          0.0                       1.0
1              9             0.211228          0.435468              0.716451            1.524050               0.189737            1.0          0.0                       0.0
2              8            -1.056767          0.435468             -0.663060           -0.480095              -0.980306            0.0          1.0                       0.0
3              5             0.143318          1.383252              2.118033           -0.447240              -0.980306            1.0          0.0                       0.0
4              5             0.971573          1.383252             -0.994142           -1.936660               0.189737            1.0          0.0                       0.0
5              8             1.769567         -0.512316              0.021177           -0.715555               1.359779            0.0          1.0                       0.0
6              6            -0.339063          0.435468             -0.994142           -0.458192               0.189737            1.0          0.0                       0.0
7              2             2.154015         -0.512316              1.157894           -0.600563               0.189737            0.0          1.0                       0.0
8              5             2.565335         -1.460099              1.687626            1.107889              -0.980306            0.0          1.0                       0.0
9              7            -0.925489          0.435468             -0.331977            1.014800              -2.150349            0.0          1.0                       0.0
10             5            -0.658132          1.383252             -0.707204            1.113365               1.359779            1.0          0.0                       0.0
11             8            -0.068995          0.435468             -0.994142           -0.419861              -0.980306            1.0          0.0                       0.0
12             8            -0.008609          0.435468             -0.663060           -1.285039              -2.150349            0.0          1.0                       0.0
13             5             0.109239         -0.512316             -0.806529           -0.677224              -0.980306            1.0          0.0                       0.0
14             2            -1.919745         -0.512316             -0.000895            0.992897               1.359779            0.0          1.0                       0.0
15             6            -1.262608          0.435468             -0.994142           -0.803168              -0.980306            1.0          0.0                       0.0
16            10            -0.579740          1.383252             -0.994142           -0.551281               0.189737            1.0          0.0                       0.0
17             7             0.198531          1.383252              0.826812           -0.387006               0.189737            1.0          0.0                       0.0
18             6             1.575575          0.435468              0.716451           -0.885305               0.189737            0.0          0.0                       1.0
19             9            -0.866334          1.383252              0.087394            1.748559               1.359779            1.0          0.0                       0.0
20             7             0.601007          0.435468              1.069605           -1.274087               0.189737            1.0          0.0                       0.0
21             6            -0.540205         -1.460099              0.098430           -0.562232              -0.980306            0.0          0.0                       1.0
22             9            -0.289667         -2.407883              2.559477           -0.868877              -2.150349            0.0          1.0                       0.0
23             3             1.107166         -0.512316             -0.663060            1.124317               1.359779            0.0          1.0                       0.0
24             7            -0.555601         -0.512316             -0.056075           -0.764837               0.189737            0.0          1.0                       0.0
25             8            -1.115742         -1.460099             -0.994142           -0.825071               0.189737            0.0          1.0                       0.0
26             8            -0.143648         -0.512316             -0.519591            1.677373               0.189737            1.0          0.0                       0.0
27             4             0.719149         -1.460099             -0.773421            0.861478               0.189737            1.0          0.0                       0.0
28             1            -1.804902         -0.512316              0.164646            1.529526               0.189737            1.0          0.0                       0.0
29             7             0.279386         -0.512316             -0.596843            1.589760               1.359779            1.0          0.0                       0.0
30             5            -0.838072          0.435468             -0.486483            0.976470               0.189737            0.0          1.0                       0.0
31             5            -0.672703         -0.512316             -0.000895            0.806720               0.189737            1.0          0.0                       0.0
32             6             1.421491         -1.460099             -0.994142           -0.737458               0.189737            0.0          1.0                       0.0
33             4            -0.436906          0.435468             -0.994142           -0.425337              -0.980306            0.0          1.0                       0.0
34             7             0.900602         -0.512316             -0.994142           -0.540329               1.359779            0.0          1.0                       0.0
35             7             0.155901         -0.512316              1.036497           -0.458192               1.359779            1.0          0.0                       0.0
36             8             0.753736          1.383252              1.974564           -0.764837               0.189737            1.0          0.0                       0.0
```



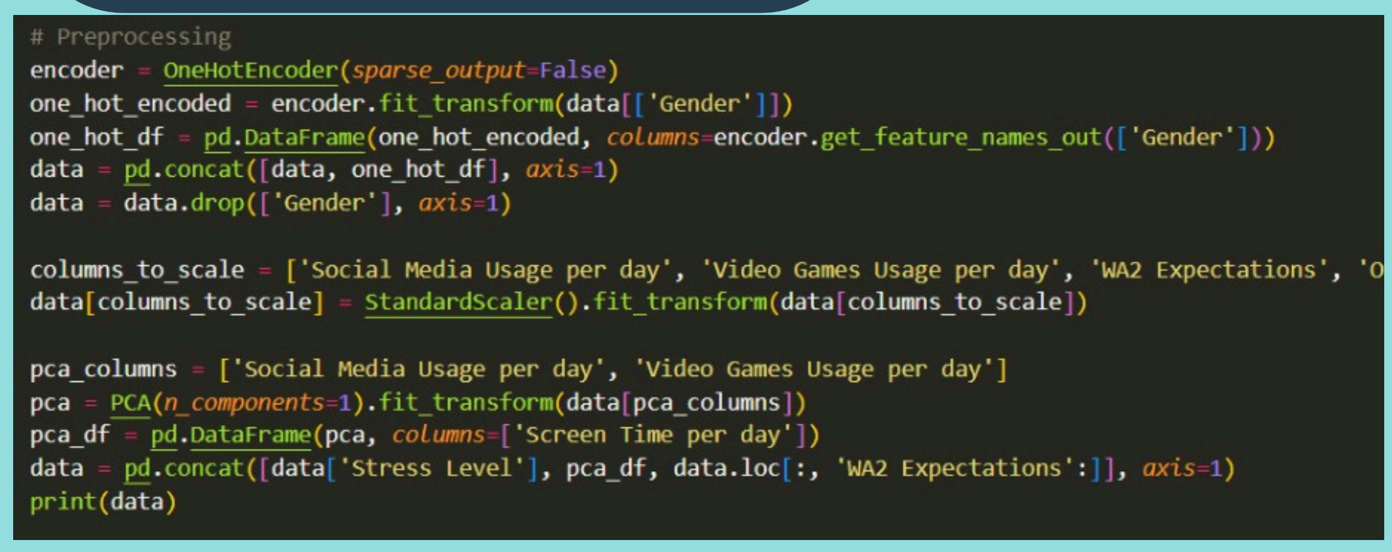
```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         2
           5       1.00      1.00      1.00        11
           6       1.00      1.00      1.00         6
           7       1.00      1.00      1.00         7
           8       1.00      1.00      1.00         4
           9       1.00      1.00      1.00         4
          10       1.00      1.00      1.00         1

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Confusion Matrix:
[[ 2  0  0  0  0  0  0  0  0  0]
 [ 0  2  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0 11  0  0  0  0  0]
 [ 0  0  0  0  0  6  0  0  0  0]
 [ 0  0  0  0  0  0  7  0  0  0]
 [ 0  0  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  0  0  0  0  4  0]
 [ 0  0  0  0  0  0  0  0  0  1]]
```

Preprocessing

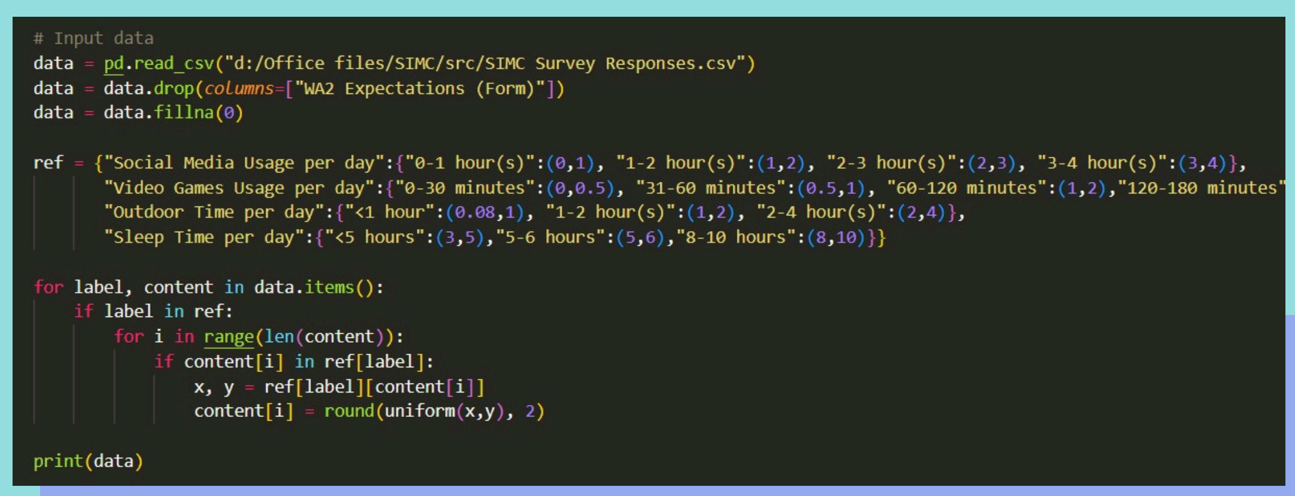


```
# Preprocessing
encoder = OneHotEncoder(sparse_output=False)
one_hot_encoded = encoder.fit_transform(data[['Gender']])
one_hot_df = pd.DataFrame(one_hot_encoded, columns=encoder.get_feature_names_out(['Gender']))
data = pd.concat([data, one_hot_df], axis=1)
data = data.drop(['Gender'], axis=1)

columns_to_scale = ['Social Media Usage per day', 'Video Games Usage per day', 'WA2 Expectations', 'O
data[columns_to_scale] = StandardScaler().fit_transform(data[columns_to_scale])

pca_columns = ['Social Media Usage per day', 'Video Games Usage per day']
pca = PCA(n_components=1).fit_transform(data[pca_columns])
pca_df = pd.DataFrame(pca, columns=['Screen Time per day'])
data = pd.concat([data['Stress Level'], pca_df, data.loc[:, 'WA2 Expectations':]], axis=1)
print(data)
```

Split data



```
# Input data
data = pd.read_csv("d:/Office files/SIMC/src/SIMC Survey Responses.csv")
data = data.drop(columns=["WA2 Expectations (Form)"])
data = data.fillna(0)

ref = {"Social Media Usage per day":{"0-1 hour(s)":(0,1), "1-2 hour(s)":(1,2), "2-3 hour(s)":(2,3), "3-4 hour(s)":(3,4)},
       "Video Games Usage per day":{"0-30 minutes":(0,0.5), "31-60 minutes":(0.5,1), "60-120 minutes":(1,2),"120-180 minutes"
       "Outdoor Time per day":{"<1 hour":(0.08,1), "1-2 hour(s)":(1,2), "2-4 hour(s)":(2,4)},
       "Sleep Time per day":{"<5 hours":(3,5),"5-6 hours":(5,6),"8-10 hours":(8,10)}}

for label, content in data.items():
    if label in ref:
        for i in range(len(content)):
            if content[i] in ref[label]:
                x, y = ref[label][content[i]]
                content[i] = round(uniform(x,y), 2)

print(data)
```

Reflection

- The model was trained pretty well according to expectation, when it is able to predict the stress level based on given factor
- What we can improve is the breadth of the data set, that is we should collect more responses so that resembling artificial sample is not nescessary.
- What we learn from the project is to create a model that can perform our prompt - and we were able to somehow train it!

References

- SIMC Workshop Resources
- Scikit-learn Resources
- The Strait Times, "Study says Singapore students suffer from high levels of anxiety"