HESS-AI Final Report

Today I am using Kaggle's Students Performance Dataset and try to analyze it to and predict the grade that the student might get.

This is my Github repository link to where I upload the code:
https://github.com/400Ping/Understanding-AI

Import Libraries

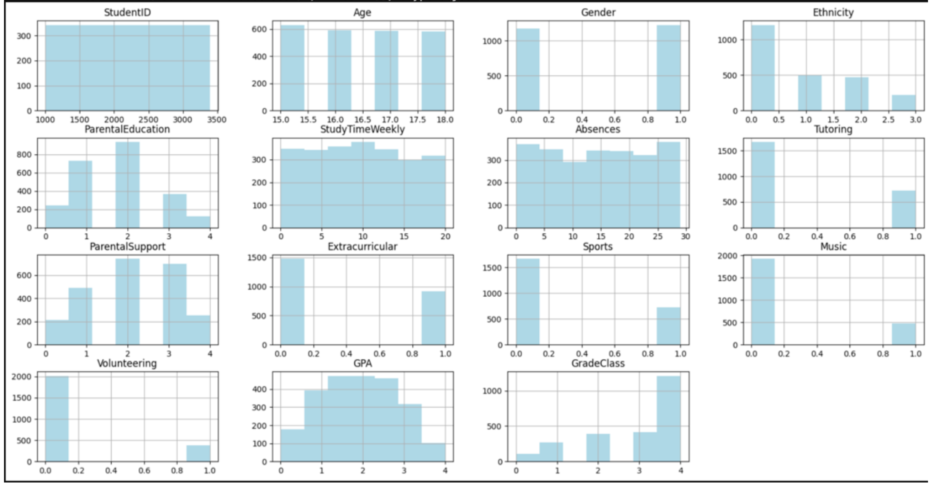
```python
# Import useful Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import RepeatedStratifiedKFold, RandomizedSearchCV, GridSearchCV, train_test_split
from scipy.stats import loguniform
```

Preprocessing Data

First, we read the csv file and look into the distribution of each feature and gain some insight into this dataset and how the features might be treated.

```python
df = pd.read_csv('Student_performance_data _.csv')
df.hist(figsize=(20,10),bins=7, color='lightblue')
```

```
array([[<Axes: title={'center': 'StudentID'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Gender'}>,
        <Axes: title={'center': 'Ethnicity'}>],
       [<Axes: title={'center': 'ParentalEducation'}>,
        <Axes: title={'center': 'StudyTimeWeekly'}>,
        <Axes: title={'center': 'Absences'}>,
        <Axes: title={'center': 'Tutoring'}>],
       [<Axes: title={'center': 'ParentalSupport'}>,
        <Axes: title={'center': 'Extracurricular'}>,
        <Axes: title={'center': 'Sports'}>,
        <Axes: title={'center': 'Music'}>],
       [<Axes: title={'center': 'Volunteering'}>,
        <Axes: title={'center': 'GPA'}>,
        <Axes: title={'center': 'GradeClass'}>, <Axes: >]], dtype=object)
```



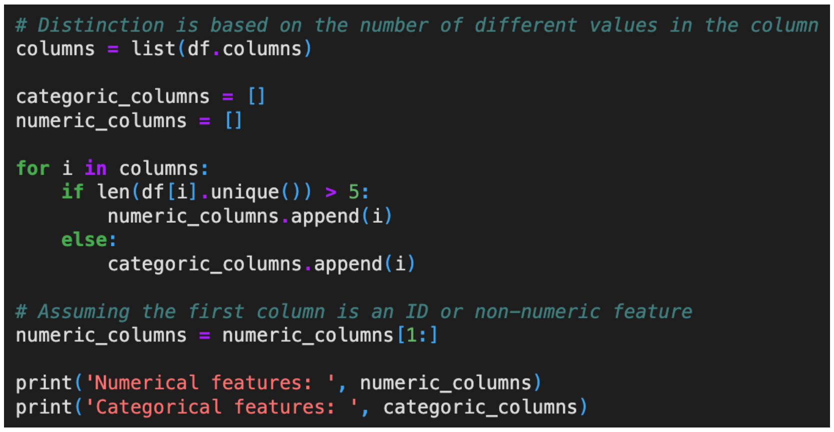
Looking at the histograms above I can make some quick inferences:

- StudentID has no distribution and logically would have no effect on Grade
- There are only 4 ages in this dataset, which strangely makes age a categorical feature
- There are only 4 ethnicity variables in this dataset
- There are a lot of low-scoring students in this dataset (a majority of 4s - Fs in GradeClass)

Define Categorical and Numerical Features

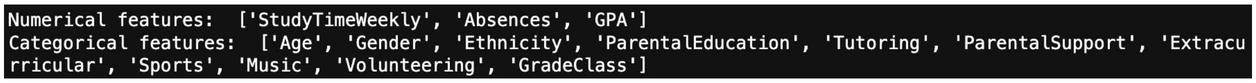
In order to proceed with assessing feature importance, there are two important steps:

1. Determine which columns are numeric and which are categoric
2. Encode the categoric columns to turn object variables into numbers
3. Scale numerical columns to ensure that large numbers have an equal effect on our model as small numbers



```python
# Distinction is based on the number of different values in the column
columns = list(df.columns)

categoric_columns = []
numeric_columns = []

for i in columns:
    if len(df[i].unique()) > 5:
        numeric_columns.append(i)
    else:
        categoric_columns.append(i)

# Assuming the first column is an ID or non-numeric feature
numeric_columns = numeric_columns[1:]

print('Numerical features: ', numeric_columns)
print('Categorical features: ', categoric_columns)
```

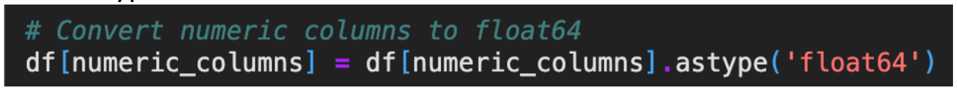
Results:



```
Numerical features:  ['StudyTimeWeekly', 'Absences', 'GPA']
Categorical features:  ['Age', 'Gender', 'Ethnicity', 'ParentalEducation', 'Tutoring', 'ParentalSupport', 'Extracu
rricular', 'Sports', 'Music', 'Volunteering', 'GradeClass']
```

In the code, I create an empty list for categoric columns and numeric columns, and then create a for loop that cycles through our features and checks the number of unique values. If there are more than 5 unique values, we can assume that the feature is numerical. If there are less than 5, we can assume it's categorical.

To ensure our numeric columns only contain numbers, I also convert the numeric columns to float64 type.



```python
# Convert numeric columns to float64
df[numeric_columns] = df[numeric_columns].astype('float64')
```

Encode Categorical Features and Scale Numerical Features

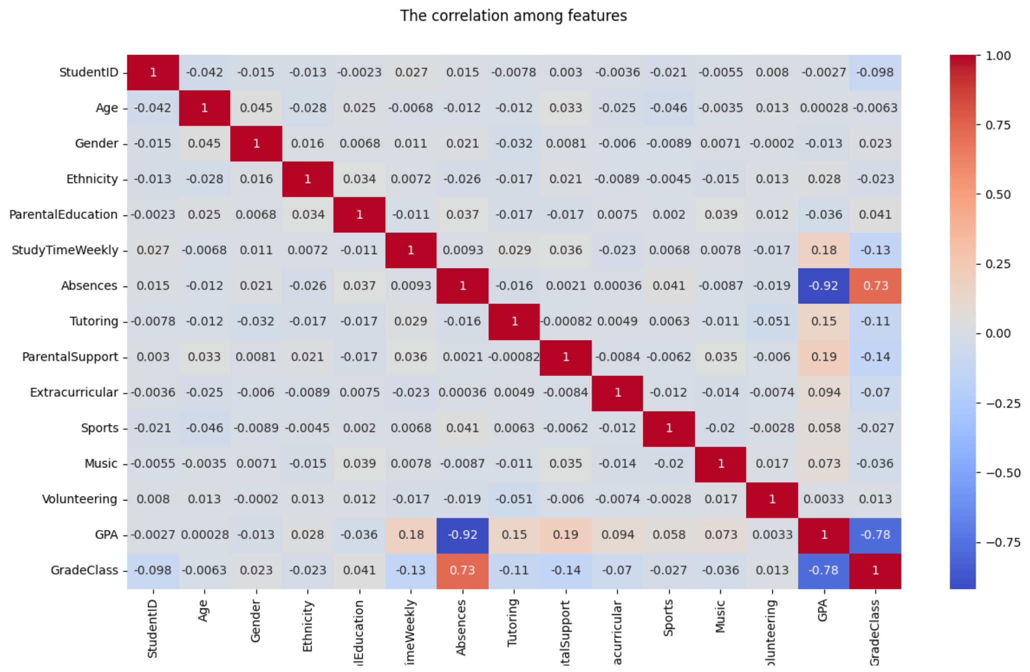
In this code I'm using LabelEncoder to encode the categoric features. I'm also using StandardScaler to scale our numeric columns to make them have equal weight on the model we use to evaluate feature importance and make predictions.

```python
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode categorical features
df = df.copy()
for column in df[categoric_columns]:
    df[column] = label_encoder.fit_transform(df[column])

# Standardize numerical features
scaler = StandardScaler()
df[numeric_columns] = scaler.fit_transform(df[numeric_columns])
```

Correlation Among Features

Check the correlation among features in the dataset.

```python
plt.figure(figsize=(16, 8))
sns.heatmap(df.corr(), annot = True, cmap = "coolwarm")
plt.title('The correlation among features',y= 1.05)
plt.show()
```



From this graph, we can see that the most correlated features to GradeClass are GPA and Absences.

Split Dataset

```python
# CHOOSE THE TARGET FEATURE HERE, IN THIS CASE IT IS 'GradeClass'
X = df.drop(columns=['GradeClass', 'GPA', 'StudentID', 'Age'])
y = df['GradeClass']

# Splitting the data into training and testing sets (e.g., 80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

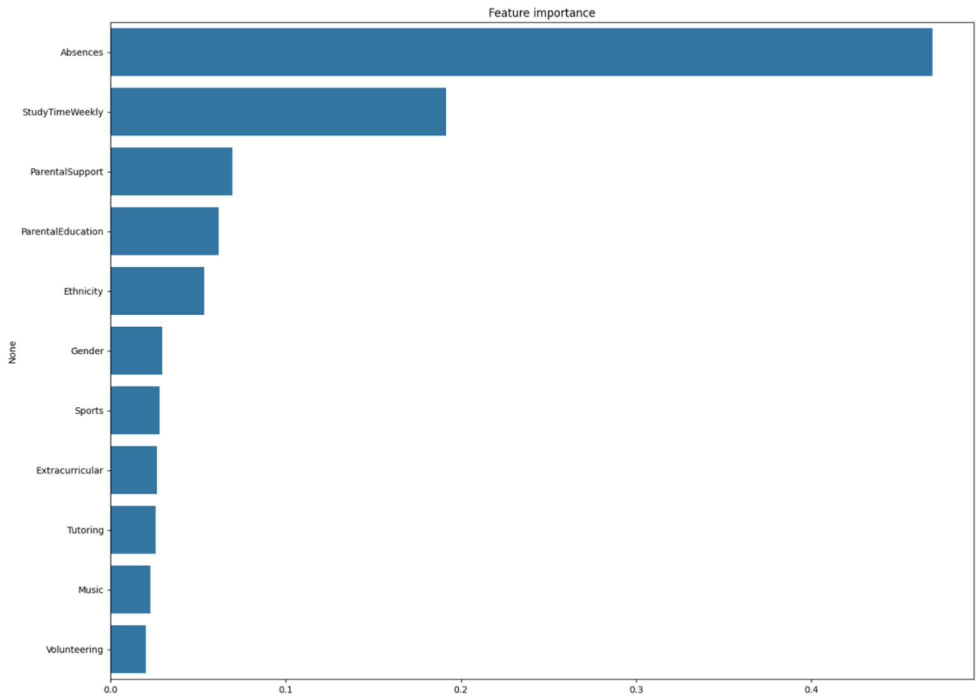
Feature Importance

In the code, I am using RandomForestClassifier to assess the relative importance of features to predicting y.

```
clf = RandomForestClassifier(random_state = 42)
clf = clf.fit(X, y)

fimp = pd.Series(data=clf.feature_importances_, index=X.columns).sort_values(ascending=False)
```

```
plt.figure(figsize=(17,13))
plt.title("Feature importance")
ax = sns.barplot(y=fimp.index, x=fimp.values, orient='h')
```

Results:

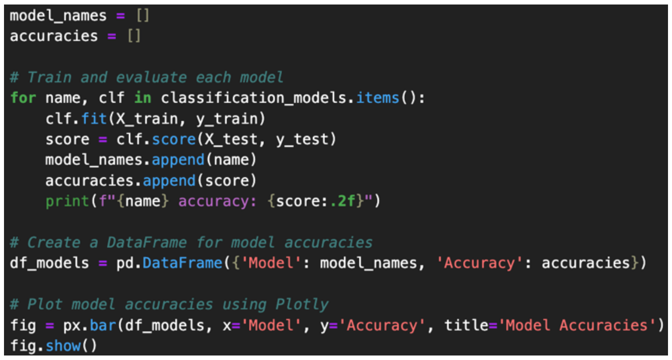


By this chart, we can see that the most important feature is absences.

Building the model

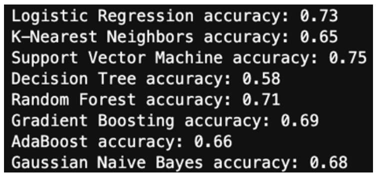
I create a dictionary of classification models and 2 empty lists - model_names and accuracies. I use a for loop to loop through the dictionary and fit the models one-by-one to the training data. I then use the fitted model to make a prediction on the y_test data and score it using clf.score function. I append model names to the model_names list and scores to the accuracies list respectively. Finally, I create a dataframe with the model_names and accuracies and plot it using a barplot to easily visualize the most accurate models.

```
# Dictionary of classification models
classification_models = {
    "Logistic Regression": LogisticRegression(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Gradient Boosting": GradientBoostingClassifier(),
    "AdaBoost": AdaBoostClassifier(),
    "Gaussian Naive Bayes": GaussianNB(),
    #"XGBoost": XGBClassifier(),
    #"CatBoost": CatBoostClassifier(silent=True),
}
```



```
model_names = []
accuracies = []

# Train and evaluate each model
for name, clf in classification_models.items():
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    model_names.append(name)
    accuracies.append(score)
    print(f"{name} accuracy: {score:.2f}")

# Create a DataFrame for model accuracies
df_models = pd.DataFrame({'Model': model_names, 'Accuracy': accuracies})

# Plot model accuracies using Plotly
fig = px.bar(df_models, x='Model', y='Accuracy', title='Model Accuracies')
fig.show()
```

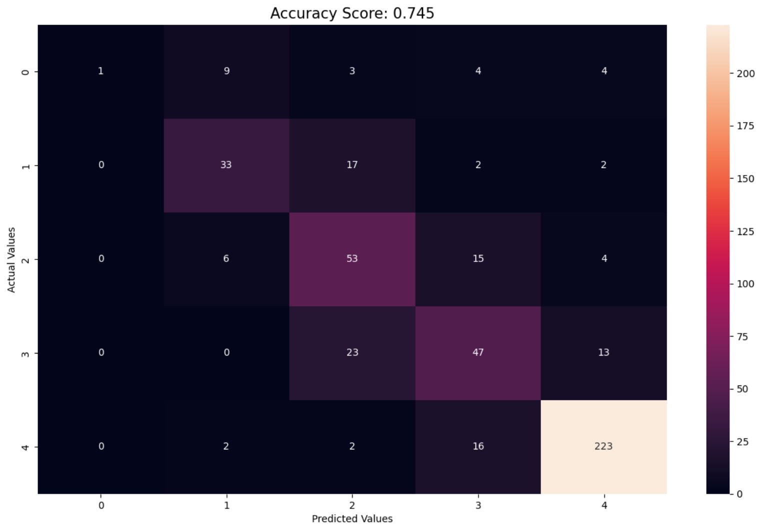
Results:



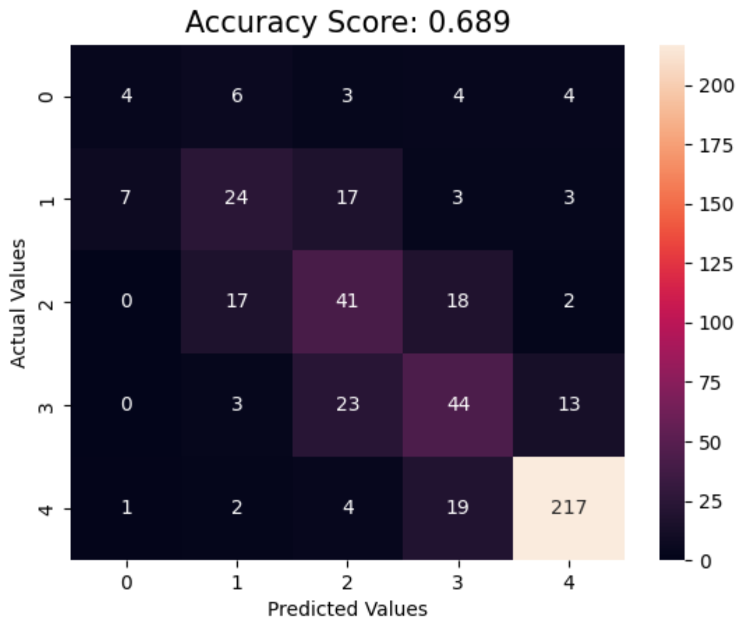
```
Logistic Regression accuracy: 0.73
K-Nearest Neighbors accuracy: 0.65
Support Vector Machine accuracy: 0.75
Decision Tree accuracy: 0.58
Random Forest accuracy: 0.71
Gradient Boosting accuracy: 0.69
AdaBoost accuracy: 0.66
Gaussian Naive Bayes accuracy: 0.68
```

We can see that the best model for this is Support Vector Machine.

Results of the Accuracy



Reduce Dimensionality Results:



By trying to analyze this data and build a model to predict the GradeClass has teach me a lot about this field. I am very new to this field and have taken a lot of references from the Kaggle's notebook. I hope I can by doing more similar practices can enhance my skills and be able to do it from scratch in the near future.

References

https://www.kaggle.com/datasets/rabieelkharoua/students-performance-dataset
https://www.kaggle.com/code/joelknapp/student-performance-analysis/notebook