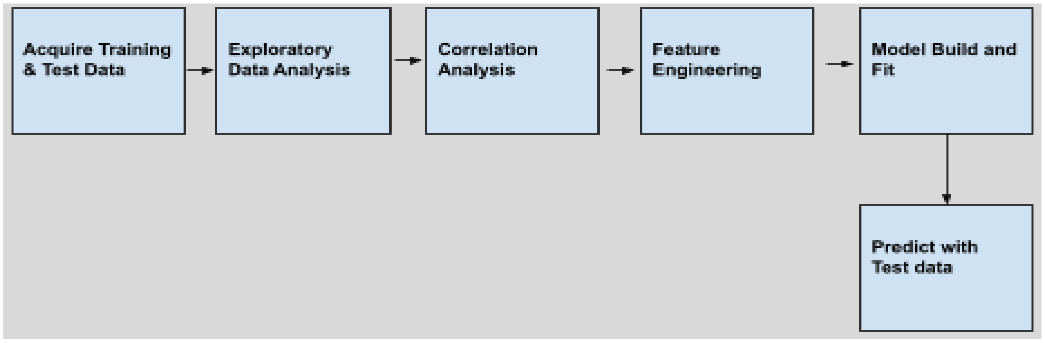
Predict Happy Score for College Data

Sreevar Rao Patiyara
https://colab.research.google.com/drive/1_OJA9UkcGYK3KrRLoeqxcJpdUjPkmr6P

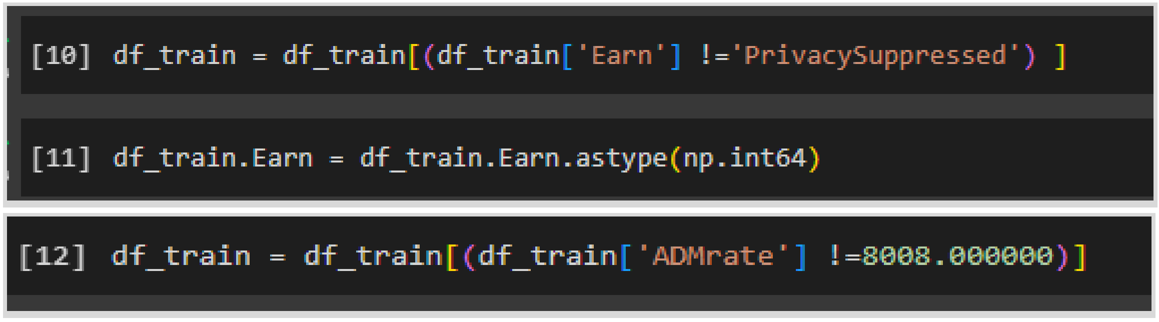
NATIONAL HIGH SCHOOL DATA SCIENCE COMPETITION BY VERITAS AI - January 2024

Project - Analyze a dataset to predict the "college happiness score" as accurately as possible, while providing an effective explanation of the methodology used.

Approach - Six step Approach with Data Acquire, EDA, Correlation, Feature Engineering, MOdel build and Predict.



Data - The training data seems to have missing values, values shifted and data type mismatch. Using the EDA analysis I have filtered out the outliers and type casted the Earn column using below code.



```
[10] df_train = df_train[(df_train['Earn'] !='PrivacySuppressed') ]

[11] df_train.Earn = df_train.Earn.astype(np.int64)

[12] df_train = df_train[(df_train['ADMrate'] !=8008.000000)]
```

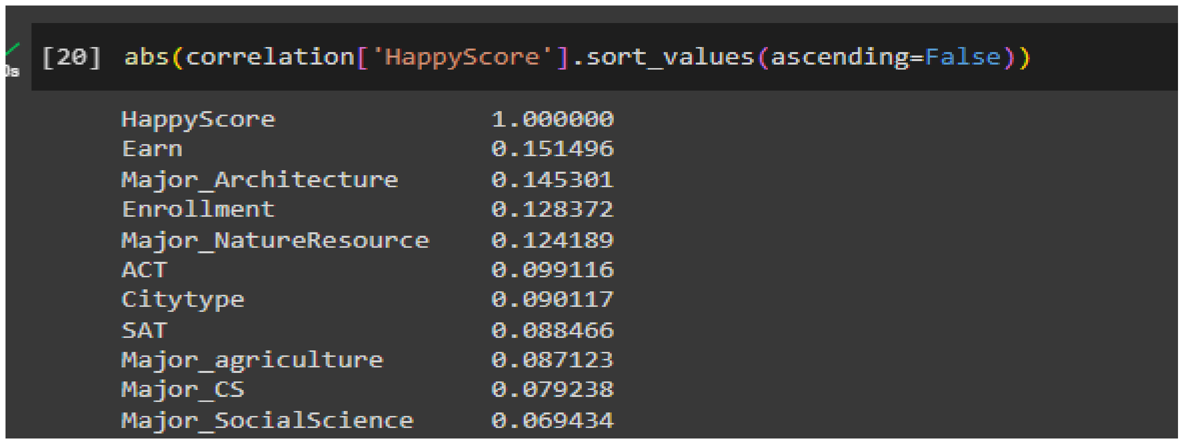
Exploratory Data Analysis -

- Check the Shape - The shape of a dataset refers to the number of rows and columns it contains. This check provides a quick overview of the dataset's size.
- Check the Data Types - Understanding the data types of each column is crucial for preprocessing and feature engineering.

- Check Missing Values - Identifying and handling missing values is essential for building robust models. The isnull() function in Pandas can be used to check for missing values, and sum().
- Check Unique Values for Categorical Columns - For categorical columns, understanding the unique values is important. This helps in recognizing the cardinality of categorical features and deciding how to encode them.
- The describe() function in the Pandas library is a powerful tool for generating summary statistics of numerical columns in a DataFrame. This function provides key metrics such as count, mean, standard deviation, minimum

Correlation Analysis - Correlation analysis is instrumental in revealing relationships between features. When features exhibit high correlations, it often indicates redundant information. In such cases, a common practice is to retain only one of the highly correlated features to streamline and simplify the model.

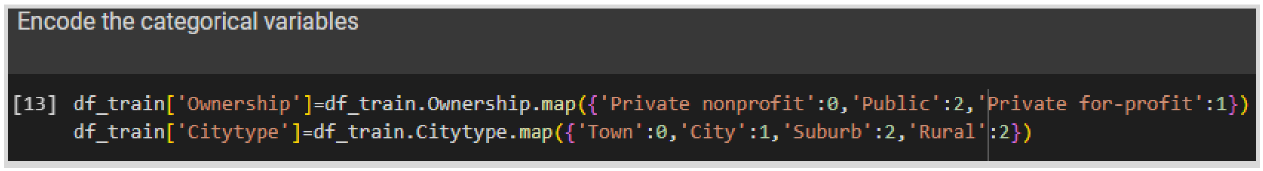
- Generate correlation table and heat map for the training data and verify the strength and direction of the relationship.
- Generate a relationship with Target Happyscore with rest of the features and look for the highest strength relationship and you can ignore the bottom ones.



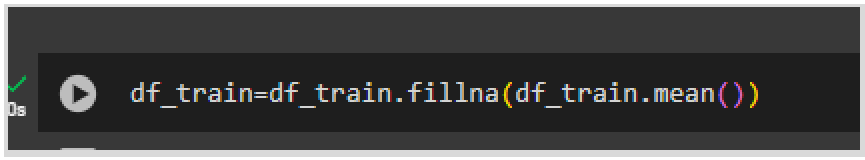
```
[20] abs(correlation['HappyScore'].sort_values(ascending=False))

     HappyScore              1.000000
     Earn                    0.151496
     Major_Architecture      0.145301
     Enrollment              0.128372
     Major_NatureResource    0.124189
     ACT                     0.099116
     Citytype                0.090117
     SAT                     0.088466
     Major_agriculture       0.087123
     Major_CS                0.079238
     Major_SocialScience     0.069434
```

Feature Engineering - Encoding categorical variables is a critical step in preparing data for machine learning models, as many algorithms require numerical input and this step converts the character values to numeric by various methods. Here I have used label encoding giving the number values to each categorical value. You can use One-hot encoding, Ordinal and binary encoding too..

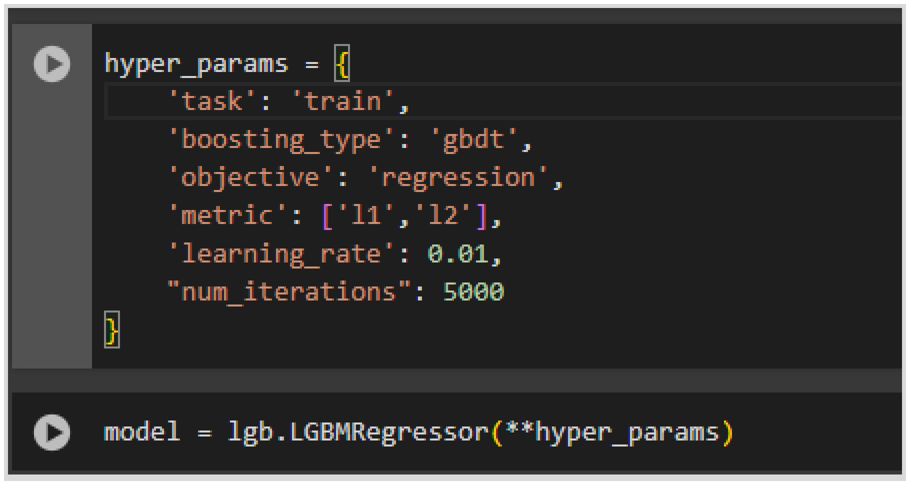

Encode the categorical variables

```
[13] df_train['Ownership']=df_train.Ownership.map({'Private nonprofit':0,'Public':2,'Private for-profit':1})
     df_train['Citytype']=df_train.Citytype.map({'Town':0,'City':1,'Suburb':2,'Rural':2})
```



```
df_train=df_train.fillna(df_train.mean())
```

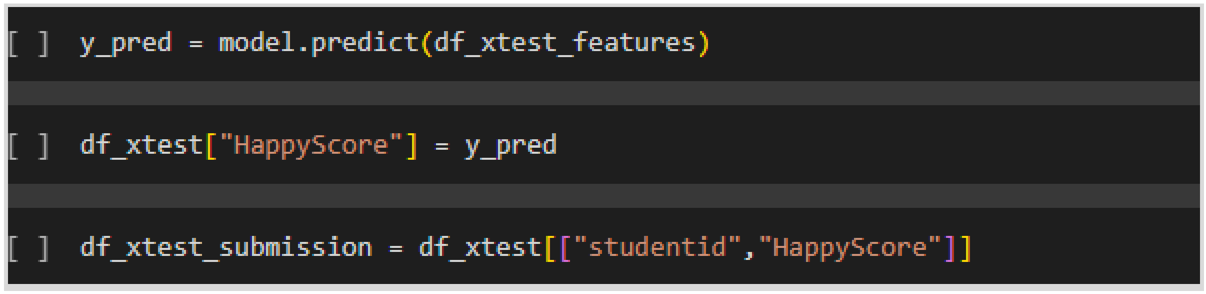
Model - I have used Light GBM Regressor with few hyper parameters. The same can be automated using Grid Search for the best possible parameters. Automating the tuning of hyperparameters using Grid Search is a common practice to find the best combination that optimizes the model's performance.



```
hyper_params = {
    'task': 'train',
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': ['l1','l2'],
    'learning_rate': 0.01,
    "num_iterations": 5000
}
```

```
model = lgb.LGBMRegressor(**hyper_params)
```

Prediction - Once the model fit is complete, read the test data and do the feature engineering similar to what we made in training data and predict the Happy Score using the LGB model. Make sure to perform the same feature engineering steps on the test data that you applied to the training data. This ensures consistency in the input format for the model.



```
[ ]  y_pred = model.predict(df_xtest_features)
```

```
[ ]  df_xtest["HappyScore"] = y_pred
```

```
[ ]  df_xtest_submission = df_xtest[["studentid","HappyScore"]]
```