GABRIEL EL HACHEM
JADE FALCK

FINAL PROJECT

SKILLCRAFT1

PYTHON FOR DATA ANALYSIS

CONTENT

01 DATA PREPROCESSING

02 DATA EXPLORATION

03 DATA SPLITTING

04 DATA MODELLING

05 CONCLUSION

01 IMPORT LIBRAIRIES

02 IMPORT DATASET

03 ENCODING

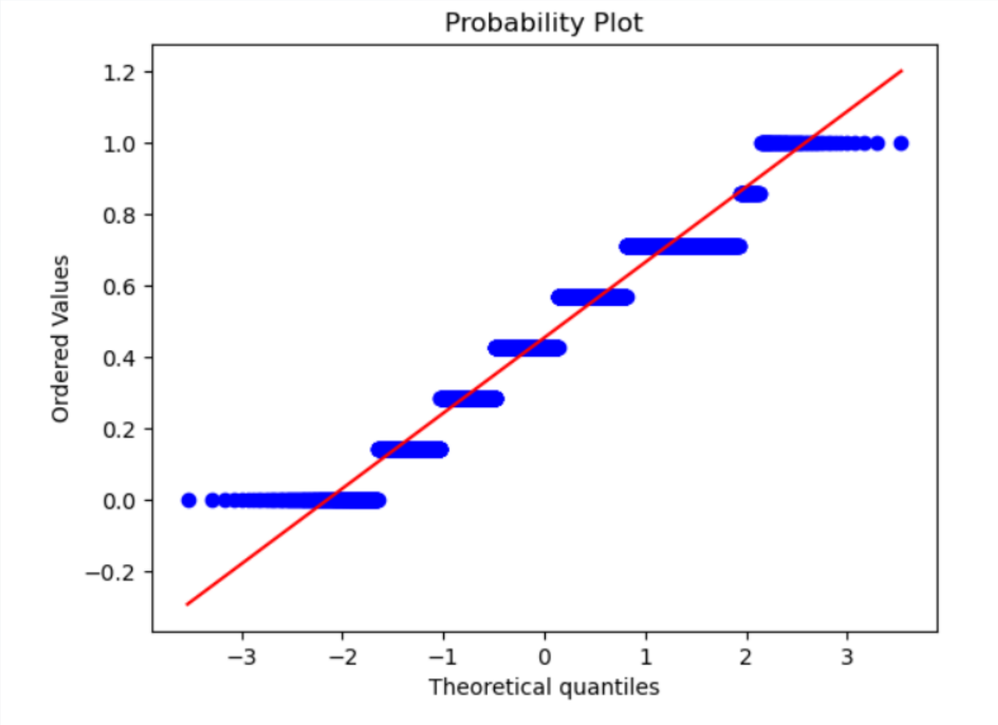
04 DROPPING COLUMNS

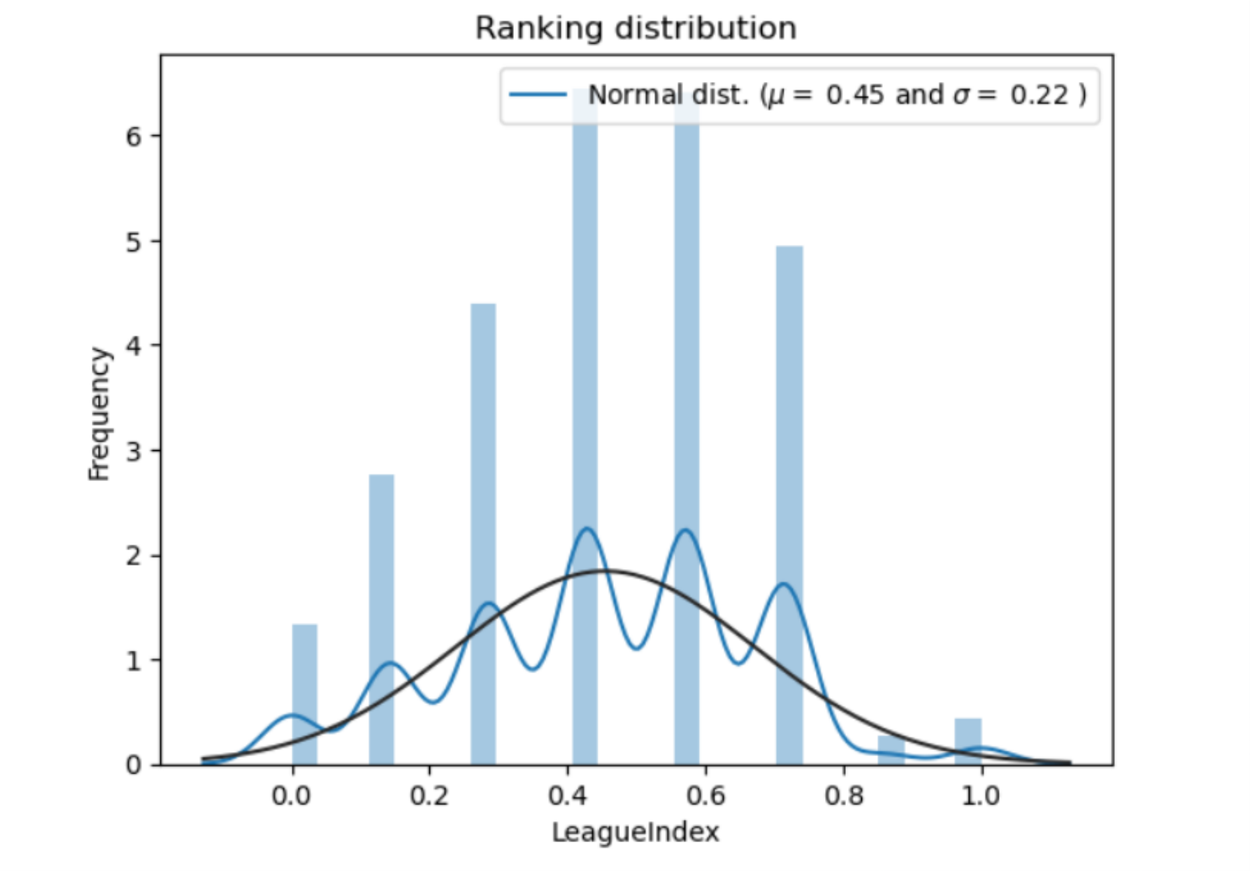
05 NORMALIZING

DATA PREPROCESSING

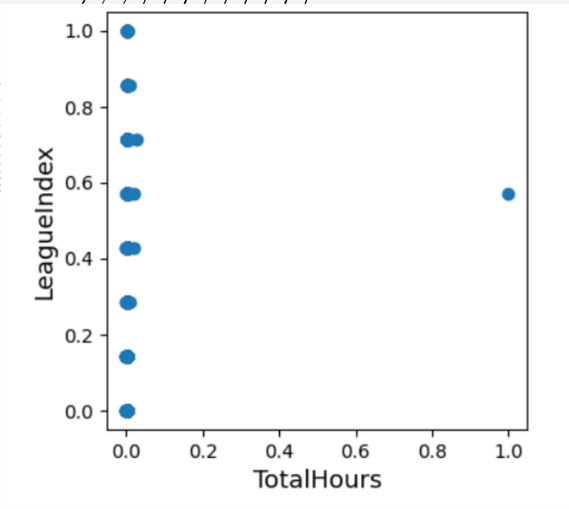
DATA EXPLORATION

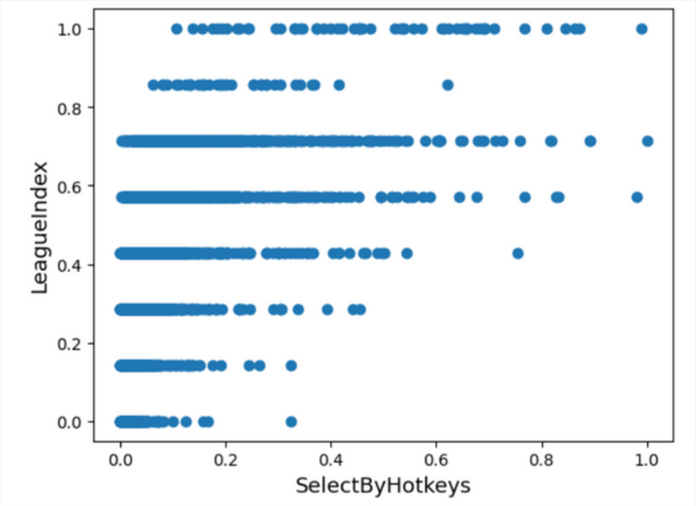
DISTRIBUTION

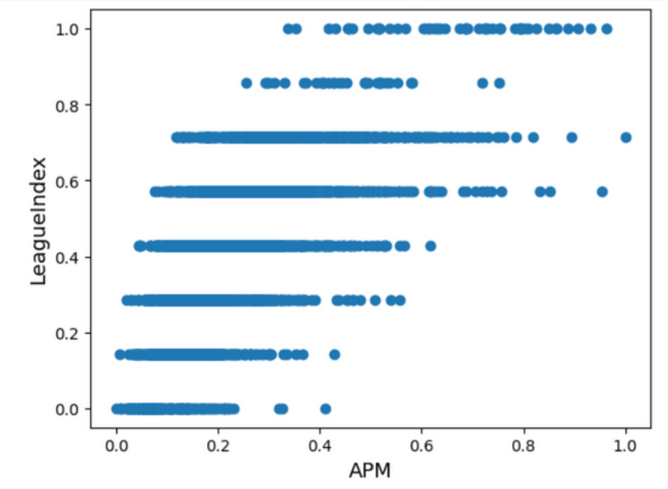


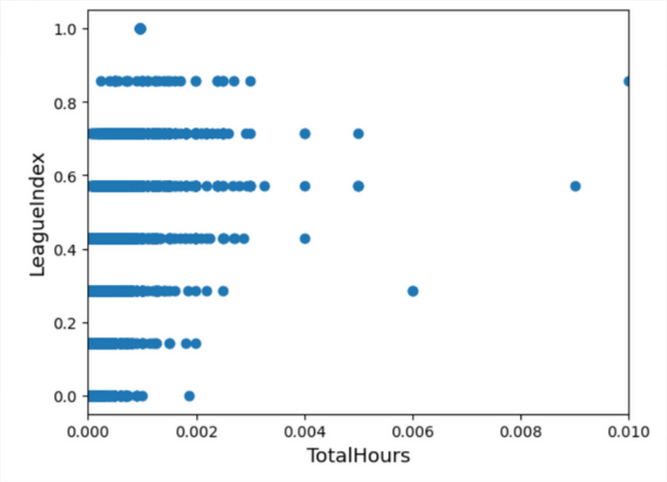


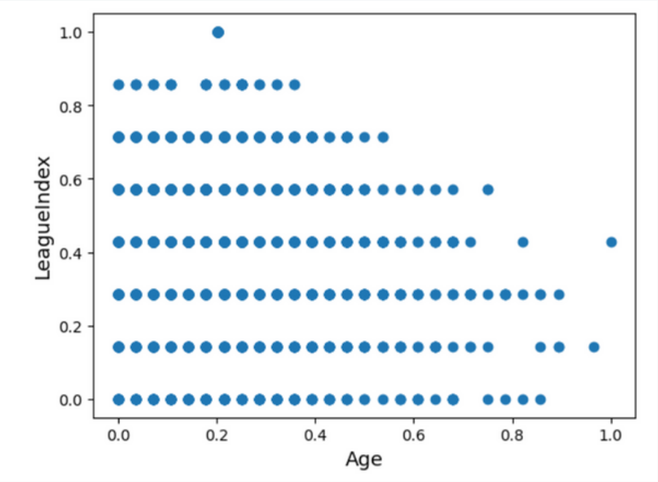
DESCRIPTIVE STATISTICS

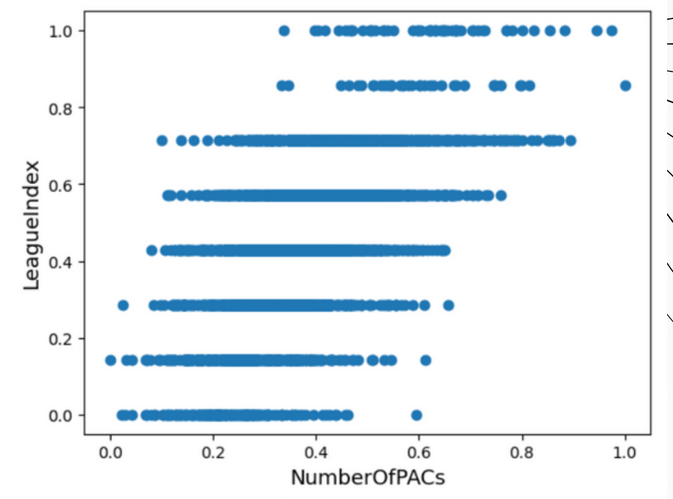




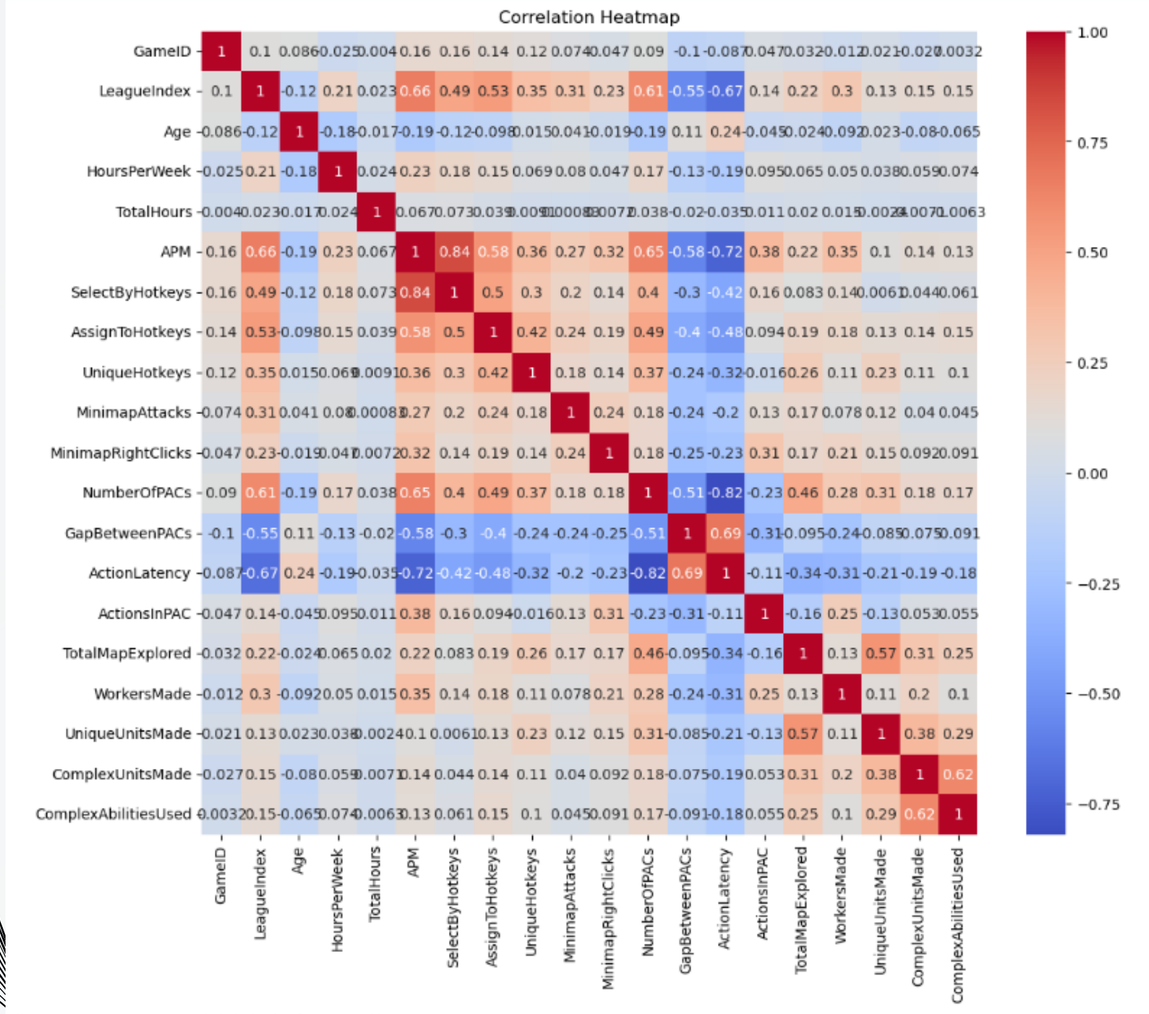


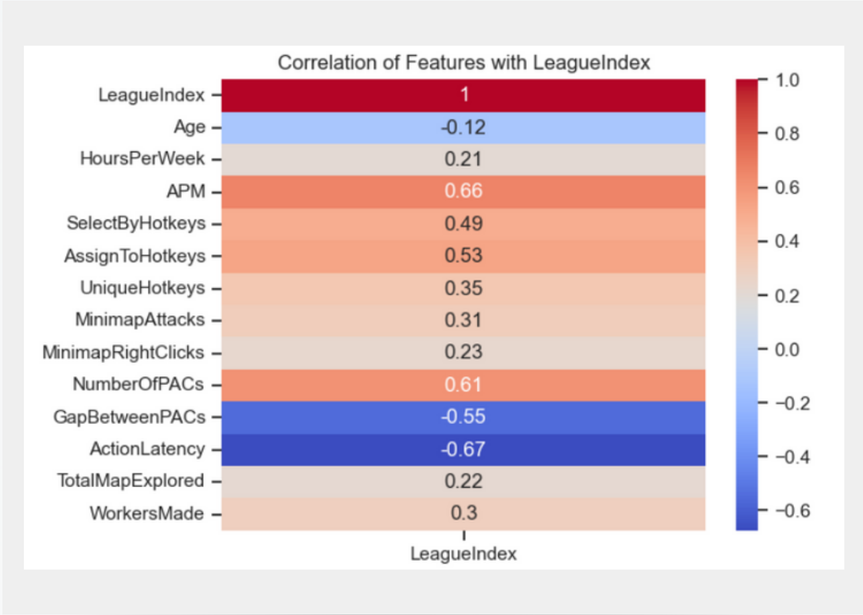


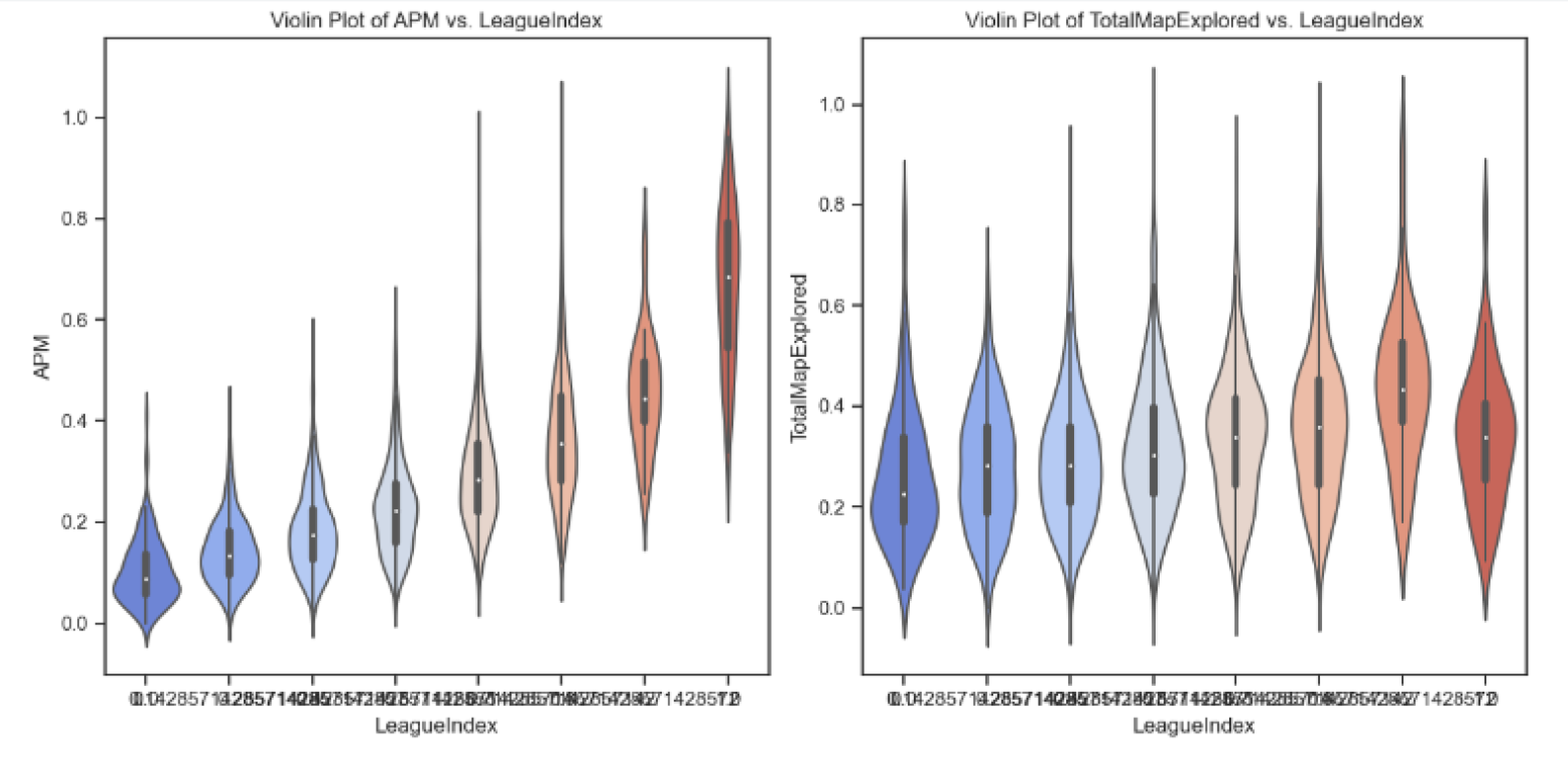




VISUALIZATIONS






Violin Plot of APM vs. LeagueIndex
APM
1.0
0.8
0.6
0.4
0.2
0.0
LeagueIndex
Violin Plot of TotalMapExplored vs. LeagueIndex
TotalMapExplored
1.0
0.8
0.6
0.4
0.2
0.0
LeagueIndex

DATA SPLITTING

```
#The target variable is 'LeagueIndex'
X = df.drop('LeagueIndex', axis=1)
y = df['LeagueIndex']
```

```
#X will contain all the features, and y will contain the target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

DATA MODELLING

Linear Regression

```python
# Define a pipeline combining a standard scaler and a linear regressor
pipeline = make_pipeline(StandardScaler(), LinearRegression())
```

Since LinearRegression doesn't have hyperparameters to tune, we don't need a grid search

```python
# Simply fit the pipeline to the training data
pipeline.fit(X_train, y_train)
```

```
Pipeline
StandardScaler
LinearRegression
```

```python
# Predicting the Test set results using the best pipeline
y_pred_lr = best_pipeline.predict(X_test)
```

```python
# Evaluate the model
mse_lr = mean_squared_error(y_test, y_pred)
r2_lr = r2_score(y_test, y_pred)
```

```python
print(f'Mean Squared Error (MSE): {mse_lr}')
print(f'R-squared (R2): {r2_lr}')
```

```
Mean Squared Error (MSE): 0.019964830750492806
R-squared (R2): 0.5543022003294026
```

Support Vector Regression

```python
# Define the pipeline steps
pipeline = make_pipeline(StandardScaler(), SVR())
```

```python
# Define a grid of hyperparameters to search
param_grid = {
    'svr__C': [0.1, 1, 10],            # Regularization parameter
    'svr__kernel': ['ldinear', 'rbf'],  # Kernel type: linear or radial basis f
    'svr__epsilon': [0.1, 0.2, 0.3]   # Epsilon parameter
}
```

```python
# Create the grid search with cross-validation
grid_search = GridSearchCV(svr_pipeline, param_grid, cv=5, scoring='neg_mean_sq
```

```python
# Fit the grid search to the training data
grid_search.fit(X_train, y_train)
```

```python
# Get the best parameters from the grid search
best_parameters = grid_search.best_params_
print("Best Parameters:", best_parameters)
```

```
Best Parameters: {'svr__C': 0.1, 'svr__epsilon': 0.1, 'svr__kernel': 'rbf'}
```

```python
# Assuming best_parameters is your dictionary with keys like 'svr__C'
fixed_parameters = {key.replace('svr__', ''): value for key, value in best_para
```

```python
# Retrieve the best pipeline
best_pipeline = grid_search.best_estimator_
```

```python
# Predicting the Test set results
y_pred_svr = svr_model.predict(X_test)
```

```python
# y_pred now contains the predicted LeagueIndex for the test set
# You can compare y_pred with y_test to evaluate the performance of your model
# For example, using mean squared error (MSE) or another suitable metric
from sklearn.metrics import mean_squared_error
```

```python
mse_svr = mean_squared_error(y_test, y_pred)
r2_svr = r2_score(y_test, y_pred)
print(f"The Mean Squared Error of the SVR model is: {mse_svr}")
print(f'R-squared (R2): {r2_svr}')
```

```
The Mean Squared Error of the SVR model is: 0.02131464064535061
R-squared (R2): 0.5241688469526466
```

Elastic Net Regression

```python
# Initialize Elastic Net model
elastic_net = ElasticNet(random_state=42)
```

```python
# Define a grid of parameters to search over
param_grid = {
    'alpha': np.logspace(-4, 0, 10),    # Regularization strength
    'l1_ratio': np.linspace(0.01, 1, 10) # L1 ratio (0=L2, 1=L1)
}
```

```python
# Grid search with cross-validation
grid_search = GridSearchCV(elastic_net, param_grid, cv=5, scoring='neg_mean_squ
grid_search.fit(X_train, y_train)
```

```
GridSearchCV
estimator: ElasticNet
ElasticNet
```

```python
# Best parameters
best_parameters = grid_search.best_params_
print("Best Parameters:", best_parameters)
```

```
Best Parameters: {'alpha': 0.046415888336127774, '
```

```python
# Train the model with the best parameters
elastic_net_best = ElasticNet(**best_parameters)
elastic_net_best.fit(X_train, y_train)
```

```
ElasticNet
ElasticNet(alpha=0.046415888336127774, l1_ratio=0.
```

```python
# Predict on the test set
y_pred_enr = elastic_net_best.predict(X_test)
```

```python
# Calculate the Mean Squared Error
mse_enr = mean_squared_error(y_test, y_pred)
r2_enr = r2_score(y_test, y_pred)
print(mse_enr)
print(f'R-squared (R2): {r2_enr}')
```

```
0.01926698373143444
R-squared (R2): 0.5698810391779774
```

```python
# Feature names
feature_names = ['Age', 'HoursPerWeek', 'APM', 'SelectByHotkeys', 'AssignToHotk
                 'UniqueHotkeys', 'MinimapAttacks', 'MinimapRightClicks', 'Numb
                 'GapBetweenPACs', 'ActionLatency', 'ActionsInPAC', 'TotalMapEx
                 'WorkersMade', 'UniqueUnitsMade', 'ComplexUnitsMade', 'Complex
```

```python
# Coefficients from the model
coefficients = elastic_net_best.coef_
```

```python
# Create a DataFrame to display feature names and their corresponding coefficie
feature_importance = pd.DataFrame({'Feature': feature_names, 'Coefficient': coe
```

```python
# Sorting the features by their coefficients
feature_importance = feature_importance.sort_values(by='Coefficient', ascending
```

DATA MODELLING

Linear Regression

```
# If you want to see the coefficier
coefficients = pd.DataFrame(linear_
print(coefficients)
```

	Coefficient
Age	0.008472
HoursPerWeek	0.010280
APM	-0.008333
SelectByHotkeys	0.037823
AssignToHotkeys	0.026422
UniqueHotkeys	0.010171
MinimapAttacks	0.027252
MinimapRightClicks	0.004118
NumberOfPACs	0.043075
GapBetweenPACs	-0.022521
ActionLatency	-0.059965
ActionsInPAC	0.012119
TotalMapExplored	-0.009273
WorkersMade	0.016577
UniqueUnitsMade	-0.007120
ComplexUnitsMade	0.004158
ComplexAbilitiesUsed	0.004906

Support Vector Regression

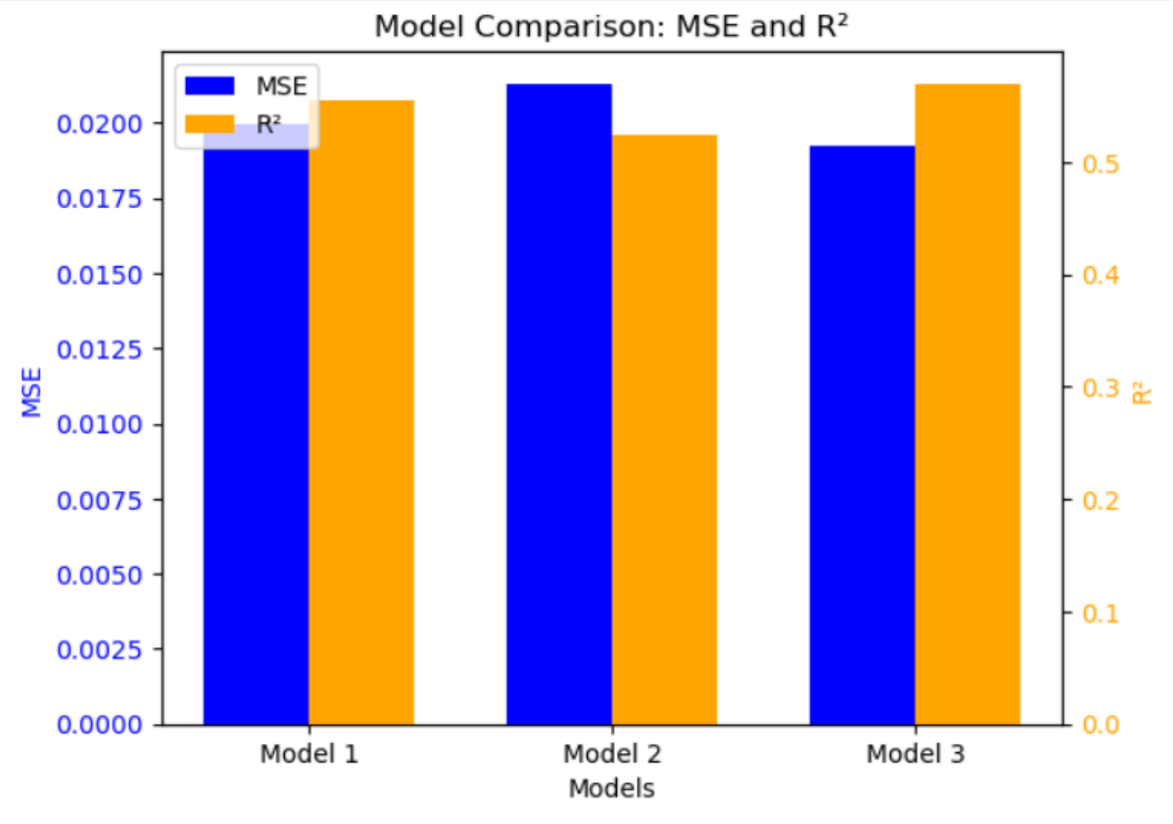
```
# Print the importances
importances = perm_importance.importances_mean
for name, importance in zip(feature_names, importances):
    print(f"{name}: {importance}")
```

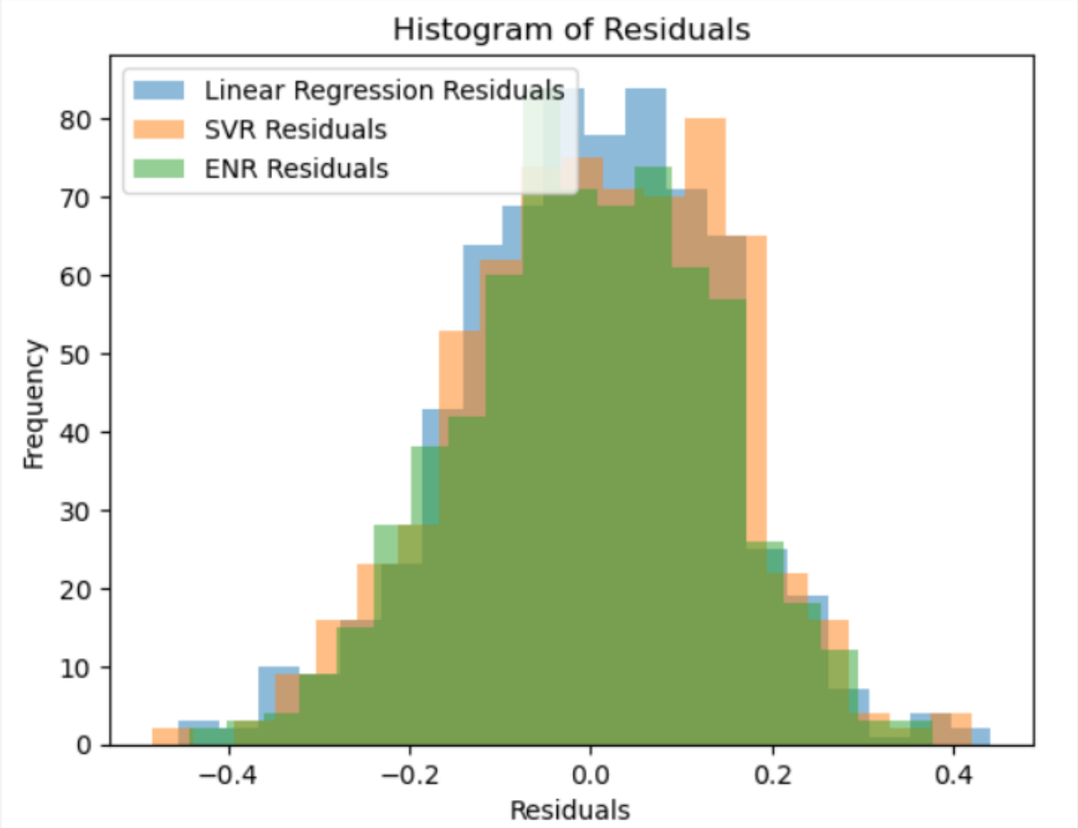
```
Age: 0.012994116693220925
HoursPerWeek: 0.011637105832684114
APM: 0.0690722385535776
SelectByHotkeys: 0.0503983987075296
AssignToHotkeys: 0.03080752728982918
UniqueHotkeys: 0.02870476945601704
MinimapAttacks: 0.03705654643595693
MinimapRightClicks: 0.00013763813707522998
NumberOfPACs: 0.029477969640387465
GapBetweenPACs: 0.04867070526257959
ActionLatency: 0.08552353980707454
ActionsInPAC: 0.019205198770492104
TotalMapExplored: 0.024288328247411763
WorkersMade: 0.022271044273435577
UniqueUnitsMade: 0.012333423660057574
ComplexUnitsMade: 0.012738493757478996
ComplexAbilitiesUsed: 0.026550523433633143
```

Elastic Net Regression

```
# Display the DataFrame
print(feature_importance)
```

	Feature	Coefficient
8	NumberOfPACs	0.031280
4	AssignToHotkeys	0.026162
6	MinimapAttacks	0.025503
3	SelectByHotkeys	0.021531
2	APM	0.020631
13	WorkersMade	0.015246
5	UniqueHotkeys	0.009732
1	HoursPerWeek	0.009702
0	Age	0.006940
16	ComplexAbilitiesUsed	0.005048
15	ComplexUnitsMade	0.003345
7	MinimapRightClicks	0.003205
11	ActionsInPAC	0.002042
14	UniqueUnitsMade	-0.005830
12	TotalMapExplored	-0.007080
9	GapBetweenPACs	-0.024265
10	ActionLatency	-0.052750


Model Comparison: MSE and R²
MSE
R²
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0.5
0.4
0.3
0.2
0.1
0.0
Model 1
Model 2
Model 3
Models


Histogram of Residuals
Linear Regression Residuals
SVR Residuals
ENR Residuals
Frequency
80
70
60
50
40
30
20
10
0
−0.4
−0.2
0.0
0.2
0.4
Residuals

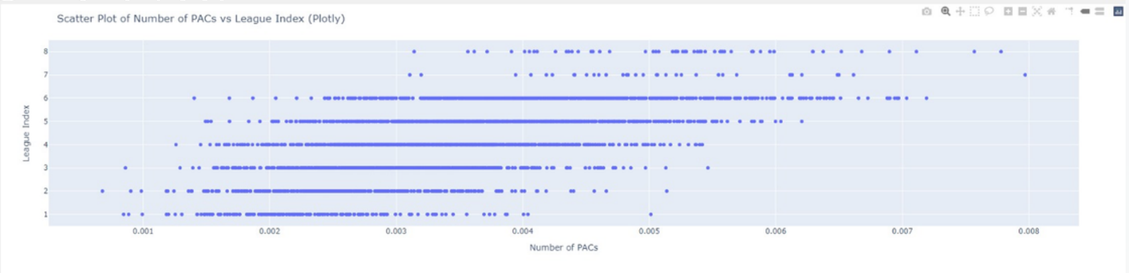
FLASK

Interactive Plot Selector

Select a plot type: Total Hours

- Total Hours
- APM
- Number Of PAC's
- Gap Between PAC's
- Select By Hotkeys
- Age

Generate Plot



Interactive Plot Selector

Select a plot type: Total Hours

Generate Plot

Models

Linear Regression

Mean Squared Error (MSE): 0.019964830750492806
R-squared (R^2): 0.5543022003294026

Linear Regression Graph

Support Vector Regression (SVR)

Best Parameters: {'svr__C': 0.1, 'svr__epsilon': 0.1, 'svr__kernel': 'rbf'}
Mean Squared Error (MSE): 0.02131464064535061
R-squared (R^2): 0.5241688469526466

Age: 0.012994116693220925

SVR Graph

Elastic Net Regression

Best Parameters: {'alpha': 0.046415888336127774, 'l1_ratio': 0.01}
Mean Squared Error (MSE): 0.01926698373143444
R-squared (R^2): 0.5698810391779774

Elastic Net Graph

Model Comparison: MSE and R^2

Model Comparison Graph

PYTHON

app.py

```python
from flask import Flask, render_template, request
import pandas as pd
import plotly.express as px
import matplotlib as plt

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/plot', methods=['GET', 'POST'])
def plot():
    if request.method == 'POST':
        selected_plot = request.form['plot_type']
    else:
        selected_plot = 'scatter_matplotlib'  # Valeur par défaut

    # Charger le dataset
    df = pd.read_csv('data/SkillCraft1_Dataset.csv')

    if selected_plot == 'TotalHours':
        fig = px.scatter(df, x='TotalHours', y='LeagueIndex',
                         labels={'TotalHours': 'Total Hours', 'LeagueIndex': 'League Index'})
        fig.update_layout(title='Scatter Plot of Total Hours vs League Index (Plotly)')
        graph_json = fig.to_json()
    elif selected_plot == 'APM':
        # Créer un graphique Plotly
        fig = px.scatter(df, x='APM', y='LeagueIndex', labels={'APM': 'APM', 'LeagueIndex': 'League Index'})
        fig.update_layout(title='Scatter Plot of APM vs League Index (Plotly)')
        graph_json = fig.to_json()
    elif selected_plot == 'NumberOfPACs':
        fig = px.scatter(df, x='NumberOfPACs', y='LeagueIndex',
                         labels={'NumberOfPACs': 'Number of PACs', 'LeagueIndex': 'League Index'})
        fig.update_layout(title='Scatter Plot of Number of PACs vs League Index (Plotly)')
        graph_json = fig.to_json()
    elif selected_plot == 'GapBetweenPACs':
        fig = px.scatter(df, x='GapBetweenPACs', y='LeagueIndex',
                         labels={'GapBetweenPACs': 'Gap Between PACs', 'LeagueIndex': 'League Index'})
        fig.update_layout(title='Scatter Plot of Gap Between PACs vs League Index (Plotly)')
```

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flask Plotly Example</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f0f0f0;
            color: #333;
            text-align: center;
            margin: 20px;
        }

        h1 {
            color: #1f77b4;
        }

        form {
            margin-top: 20px;
        }

        label, select, input {
            font-size: 16px;
            margin-bottom: 10px;
            margin-right: 10px;
        }

        select {
            padding: 5px;
            width: 200px;
        }

        input[type="submit"] {
            background-color: #1f77b4;
            color: #fff;
            padding: 10px;
            border: none;
```

graph.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Graph</title>
    <!-- Charger la bibliothèque Plotly -->
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f0f0f0;
            color: #333;
            text-align: center;
            margin: 20px;
        }

        #graph {
            margin-top: 20px;
        }

        .description {
            text-align: left;
            margin-top: 20px;
            display: none;
        }
    </style>
</head>
<body>
    <!-- Emplacement pour le graphique interactif -->
    <div id="graph"></div>

    <div class="description" id="TotalHours">
        <p>Description for Total Hours vs League Index: This graph shows the relationship between the total hours played and the league index. Analyze the trends and patterns in the data to draw ins
    </div>
    <div class="description" id="APM">
        <p>Description for APM vs League Index: This graph illustrates the correlation between Actions Per Minute (APM) and the league index. Examine how APM impacts the league index.</p>
    </div>
    <div class="description" id="NumberOfPACs">
        <p>Description for Number of PACs vs League Index: Explore the connection between the number of Performed Actions Commands (PACs) and the league index. Analyze the significance of PACs in th
```

CONCLUSION