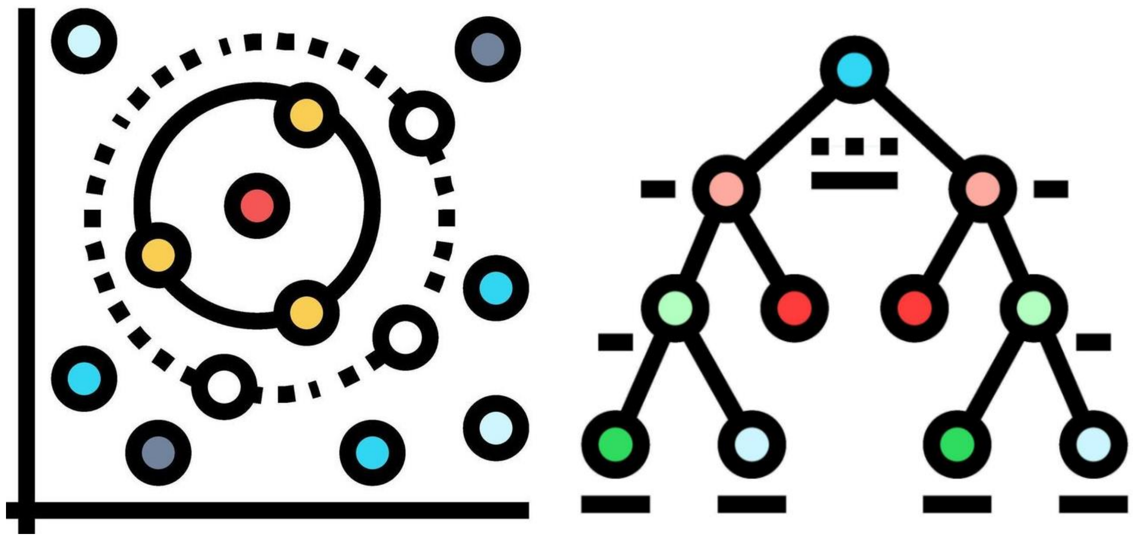
Personality-Driven Experiences: ML Classification Case Study



JUNAID

Research Scholar

M.Phil-Data Science

Table of Contents

- **1. Introduction..3**
- 1.1. Project Overview
- **2. Data Understanding and Preprocessing..4**
- 2.1. Data Description
- 2.2. Class Balance Check (Binary Variables)
- 2.3. Categorical Variables
- 2.4. Numerical Variables
- 2.5. Missing Values
- 2.6. Imputation Strategy
- 2.7. Duplicates
- 2.8. Outliers (Boxplot)
- **3. Hypothesis Testing and Feature Analysis..10**
- 3.1. Overview of Hypothesis Testing
- 3.2. Shapiro-Wilk Test (Normality)
- 3.3. Levene's Test (Equality of Variances)
- 3.4. Non-Parametric Hypothesis Testing (Introduction)
- 3.5. Mann-Whitney U Test
- 3.6. Chi-Squared Test
- 3.7. One-Hot Encoding
- 3.8. Correlation Heat-Map
- 3.9. Feature Engineering
- 3.10. Feature Selection
- 3.11. Multicollinearity: Variance Inflation Factor (VIF)
- **4. Model Evaluation...16**
- 4.1. Baseline Classifiers
- 4.1.1. Decision Tree
- 4.1.2. K-Nearest Neighbors (KNN)
- 4.1.3. Logistic Regression
- 4.1.4. Random Forest
- 4.2. Results (Before Tuning)
- 4.3. Feature Importance
- 4.4. Hyperparameter Tuning (GridSearchCV)
- 4.5. Best Parameters
- 4.6. Results (After Tuning)
- 4.7. Model Performance Comparison
- 4.8. ROC Curve

Introduction

Powering Hyper-Personalized Experiences with Personality Inference

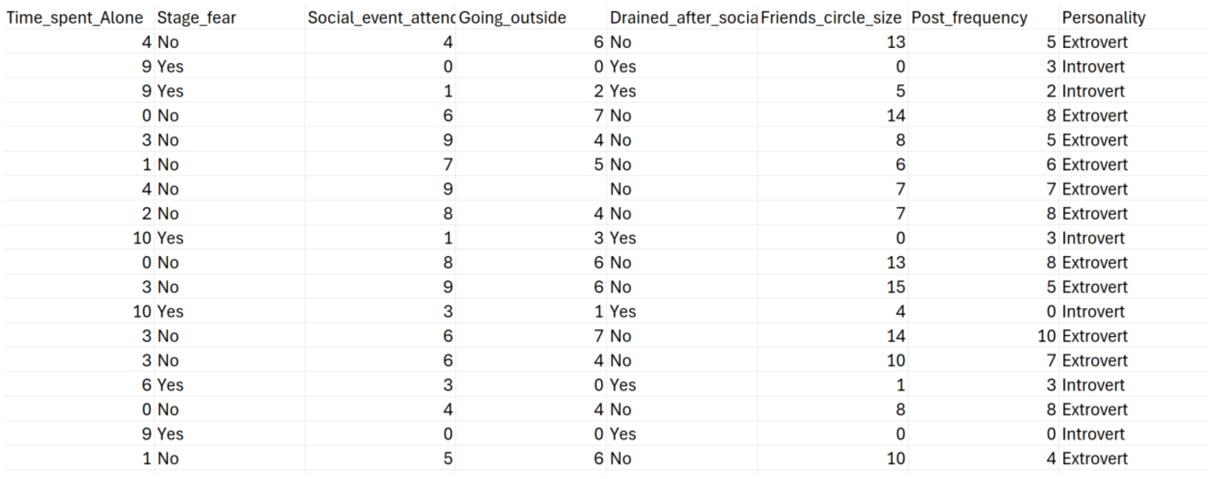
ConnectSphere, a growing social-wellness platform, is launching "Personality-Driven Experiences." This feature aims to **personalize user interactions** by inferring whether a user is an **Introvert or Extrovert** based solely on their in-app behavior. This shift will revolutionize engagement by tailoring activity suggestions, in-app challenges, and social nudges.

The core business problem is to build a robust classifier that accurately infers personality from user behavior, without explicit quizzes. This capability is vital for boosting **engagement, improving retention**, and driving **premium upsells** through hyper-personalized recommendations.

Challenges include the absence of direct personality data, potential data sparsity, the need for real-time inference, and ensuring a measurable impact on key metrics like click-through rates and premium conversions.

This report details the project objective: to **build and validate a machine learning classification model** for user personality (Introvert vs. Extrovert) using behavioral data. This model will power an **Activity Suggestion Engine** (low-intensity for introverts, group events for extroverts) and **Engagement Nudges** (e.g., outdoor activity prompts for introverted, low-activity users). Subsequent sections will cover dataset preprocessing, modeling tasks, and performance evaluation.

Data Description



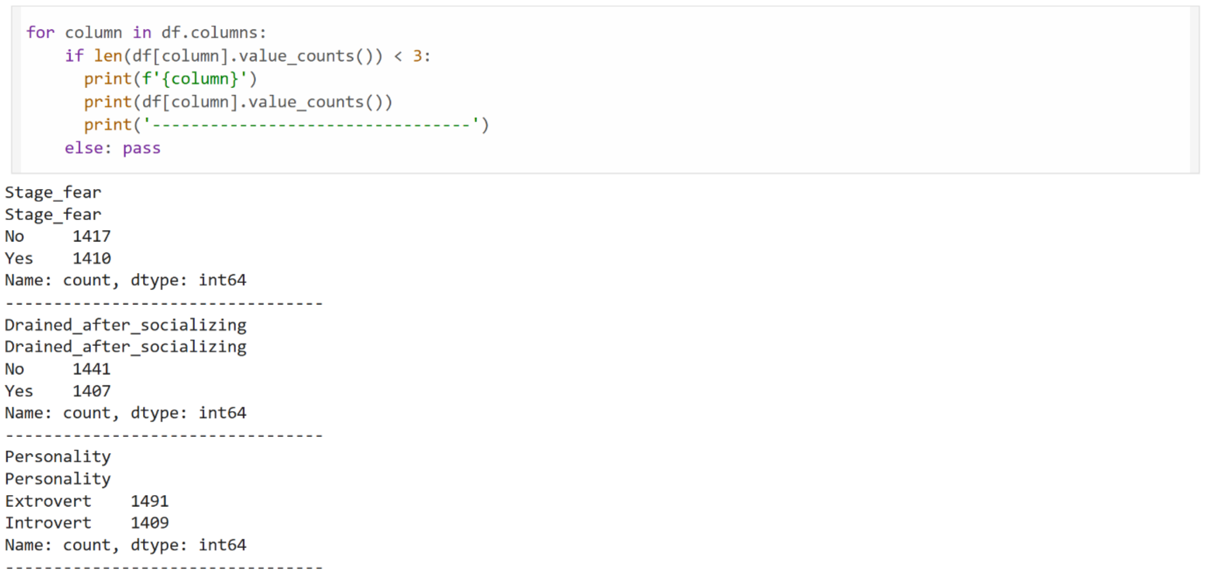
Time_spent_Alone	Stage_fear	Social_event_attenc	Going_outside	Drained_after_socia	Friends_circle_size	Post_frequency	Personality
4	No	4	6	No	13	5	Extrovert
9	Yes	0	0	Yes	0	3	Introvert
9	Yes	1	2	Yes	5	2	Introvert
0	No	6	7	No	14	8	Extrovert
3	No	9	4	No	8	5	Extrovert
1	No	7	5	No	6	6	Extrovert
4	No	9		No	7	7	Extrovert
2	No	8	4	No	7	8	Extrovert
10	Yes	1	3	Yes	0	3	Introvert
0	No	8	6	No	13	8	Extrovert
3	No	9	6	No	15	5	Extrovert
10	Yes	3	1	Yes	4	0	Introvert
3	No	6	7	No	14	10	Extrovert
3	No	6	4	No	10	7	Extrovert
6	Yes	3	0	Yes	1	3	Introvert
0	No	4	4	No	8	8	Extrovert
9	Yes	0	0	Yes	0	0	Introvert
1	No	5	6	No	10	4	Extrovert

The raw dataset will include the following key features, collected daily for each user:

- **Time_spent_Alone (hours/day):** Represents the number of hours a user spends in solitary activities within the app.
- **Comfort_Level_with_new_situations (fear_stage: Yes/No):** Indicates a user's comfort level when encountering new situations or features within the platform. 'Yes' suggests discomfort/fear, while 'No' suggests comfort.
- **Weekly_Social_Events_Attended:** The cumulative number of social events a user attends within a week.
- **Days_Going_Outdoors:** The number of days a user reports going outdoors within a week.
- **Energy_After_Socializing (drained: Yes/No):** Reflects a user's energy levels post-socialization. 'Yes' indicates feeling drained, while 'No' indicates feeling energized or neutral.
- **Size_of_Close-Friend_Circle:** The reported or inferred size of a user's close-friend circle within the platform.
- **Weekly_Social-Media_Posts:** The total number of social media posts made by a user on the platform per week.
- **Personality (Target Variable):** The inferred personality type, categorized as 'Introvert' or 'Extrovert'. This is the target variable for our classification model.

Class Balance Check

Categorical

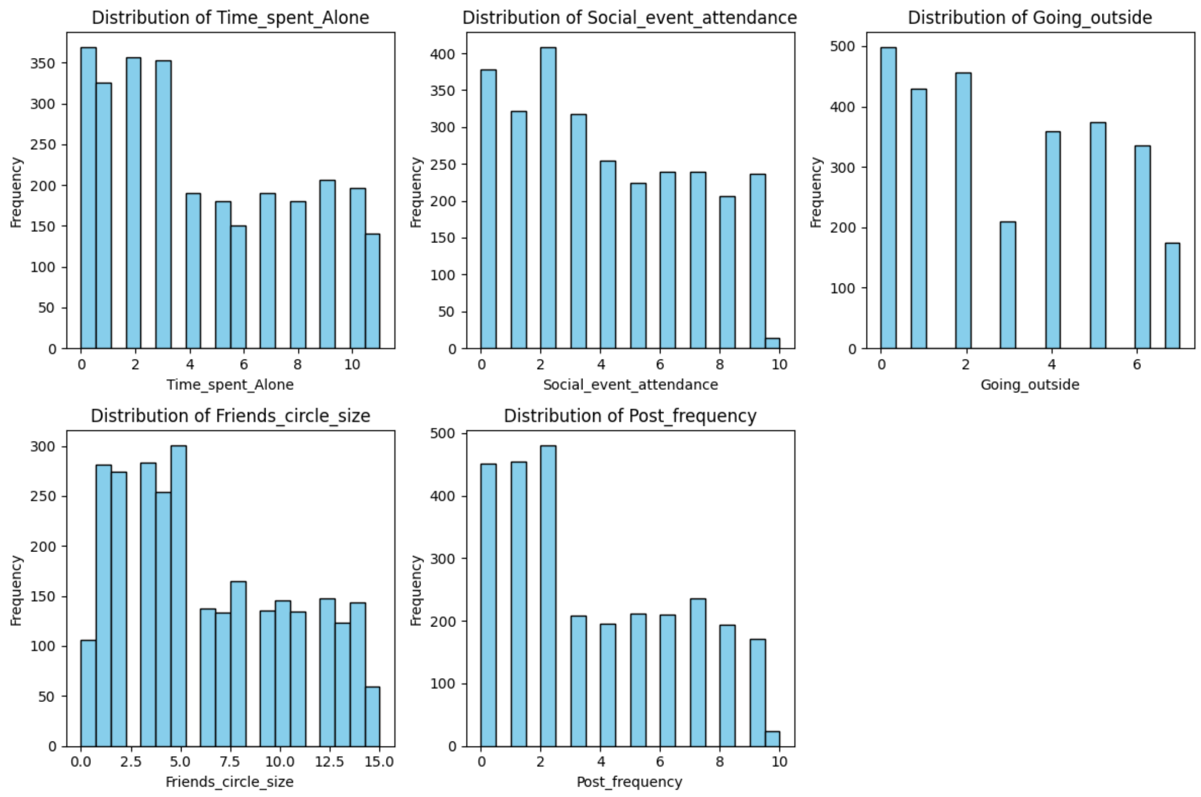


```python
for column in df.columns:
    if len(df[column].value_counts()) < 3:
      print(f'{column}')
      print(df[column].value_counts())
      print('--------------------------------')
    else: pass
```

```
Stage_fear
Stage_fear
No     1417
Yes    1410
Name: count, dtype: int64
---------------------------------
Drained_after_socializing
Drained_after_socializing
No     1441
Yes    1407
Name: count, dtype: int64
---------------------------------
Personality
Personality
Extrovert    1491
Introvert    1409
Name: count, dtype: int64
---------------------------------
```

An initial check was performed on the binary variables, including 'Comfort_Level_with_new_situations', 'Energy_After_Socializing', and the target variable 'Personality'. The distribution of classes for these features appears relatively balanced, indicating no significant skew that would necessitate advanced handling like oversampling or undersampling at this stage. For instance, 'Comfort_Level_with_new_situations' has approximately 1417 'No' entries and 1410 'Yes' entries, 'Energy_After_Socializing' has 1441 'No' and 1407 'Yes', and 'Personality' shows 1491 'Extrovert' and 1409 'Introvert' instances.

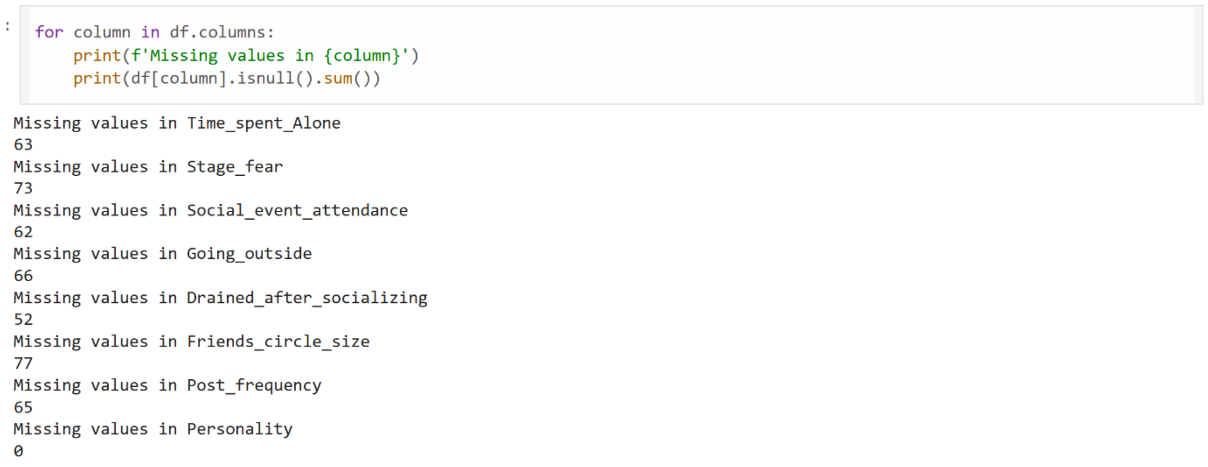
Numerical Variables



- **Time_spent_Alone:** The distribution shows a higher frequency for lower values (0-2 hours), with a gradual decrease as time spent alone increases, suggesting that most users spend less time alone within the app. This feature exhibits a **right-skewed** distribution.
- **Weekly_Social_Events_Attended:** This feature also displays a higher frequency for lower values, with frequencies declining significantly for higher event attendance numbers. This indicates a **right-skewed** distribution.
- **Days_Going_Outdoors:** Similar to social events, the distribution is **right-skewed**, indicating that many users report going outdoors for fewer days per week, with a sharp drop-off for more frequent outdoor activities.
- **Size_of_Close-Friend_Circle:** This distribution appears somewhat multimodal or spread out, with noticeable peaks at lower to mid-range friend circle sizes (e.g., around 0-2.5 and 5-7.5), and then tapering off for larger circles. While not as strongly skewed as the others, it tends towards a **positive skew**.
- **Weekly_Social-Media_Posts:** This feature is heavily right-skewed, with a very high frequency of users making few to no social media posts. The number of posts drops sharply as the frequency of posting increases, indicating a **strong right-skewness**.

These distributions provide insights into user behavior patterns and will inform potential transformations (e.g., logarithmic transformation for heavily skewed features) during feature engineering to improve model performance.

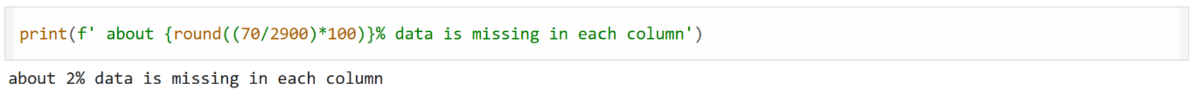
Missing Values



```
for column in df.columns:
    print(f'Missing values in {column}')
    print(df[column].isnull().sum())
```

```
Missing values in Time_spent_Alone
63
Missing values in Stage_fear
73
Missing values in Social_event_attendance
62
Missing values in Going_outside
66
Missing values in Drained_after_socializing
52
Missing values in Friends_circle_size
77
Missing values in Post_frequency
65
Missing values in Personality
0
```

- Time_spent_Alone: 63 missing values
- Comfort_Level_with_new_situations: 73 missing values
- Weekly_Social_Events_Attended: 62 missing values
- Days_Going_Outdoors: 66 missing values
- Energy_After_Socializing: 52 missing values
- Friends_circle_size: 77 missing values
- Post_frequency: 65 missing values
- Personality: 0 missing values (target variable is complete)



```
print(f' about {round((70/2900)*100)}% data is missing in each column')
```

```
about 2% data is missing in each column
```

These missing values will need to be addressed through an appropriate imputation strategy during the preprocessing phase to ensure model robustness.

Imputation Strategy

```
for col in df.columns:
    if df[col].isnull().any():
      if df[col].dtype == 'object':
          df[col].fillna(df[col].mode()[0], inplace=True)
      else:
        df[col].fillna(df[col].median(),inplace=True)
```

Categorical variables (e.g., 'Comfort_Level_with_new_situations', 'Energy_After_Socializing'), missing values will be imputed using the **mode** (most frequent value) of their respective columns.

Numerical variables (e.g., 'Time_spent_Alone', 'Weekly_Social_Events_Attended', 'Days_Going_Outdoors', 'Friends_circle_size', 'Post_frequency'), given that many exhibit skewed distributions, missing values will be imputed using the **median** of their respective columns. The median is chosen over the mean as it is less sensitive to outliers and skewness, providing a more robust imputation for these features.

Duplicates

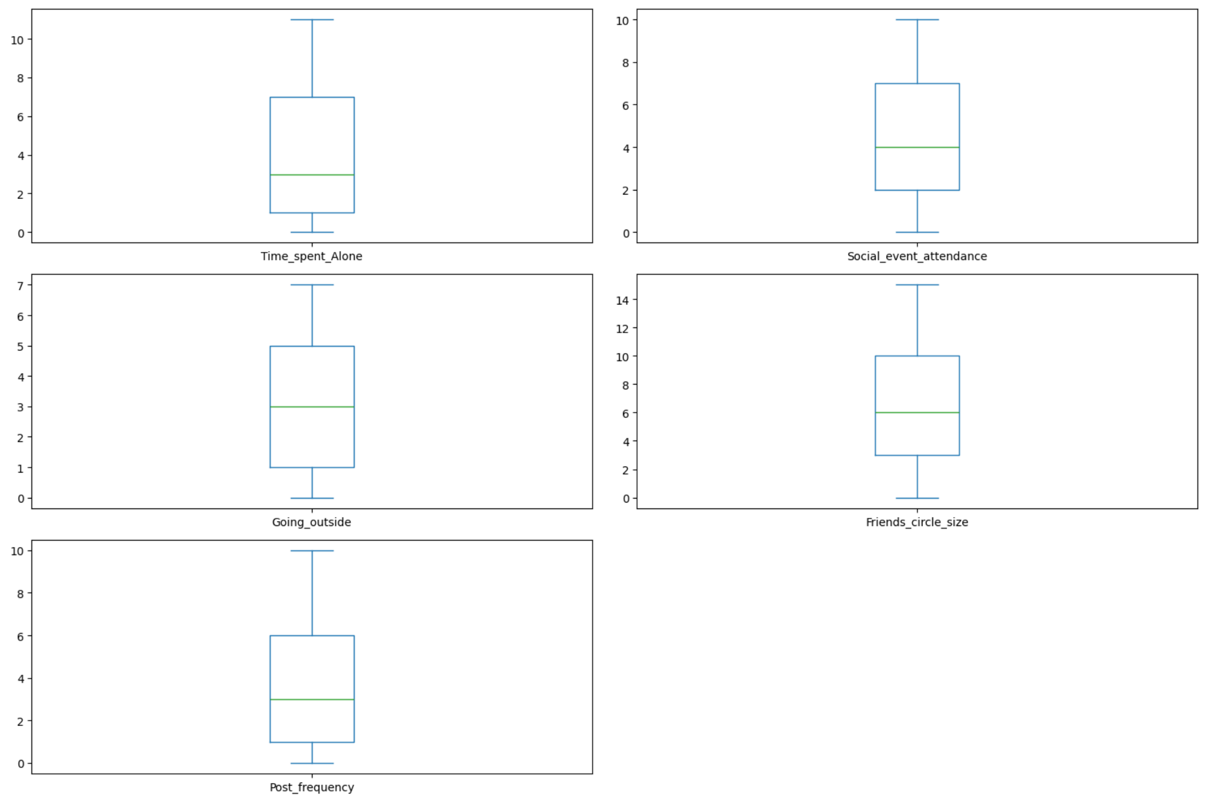
```
df.duplicated().sum()
```

```
np.int64(448)
```

```
df.drop_duplicates(inplace=True)
```

Dropping duplicate records.

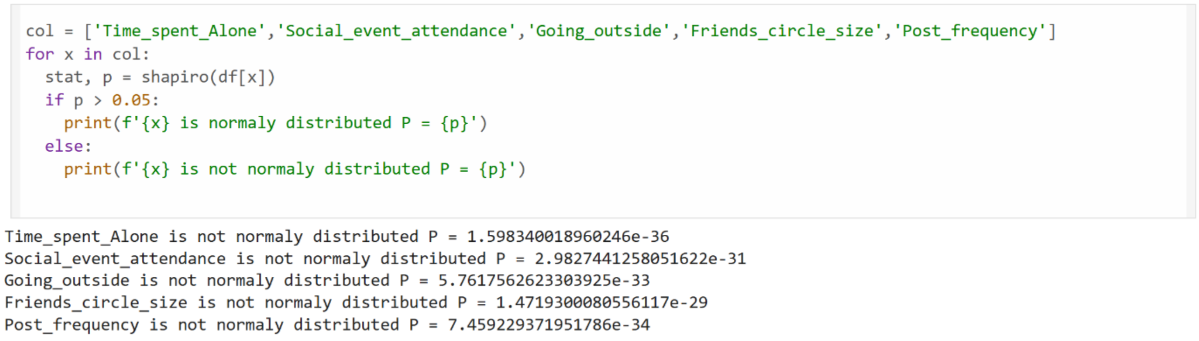
Outliers



a visual inspection of box plots for the numerical features ('Time_spent_Alone', 'Social_event_attendance', 'Going_outside', 'Friends_circle_size', and 'Post_frequency') indicates **no significant outliers** were detected. The data points appear to fall within the typical range, suggesting that no specific outlier treatment beyond median imputation for missing values will be required for these features.

Hypothesis Testing

Shapiro-Wilk

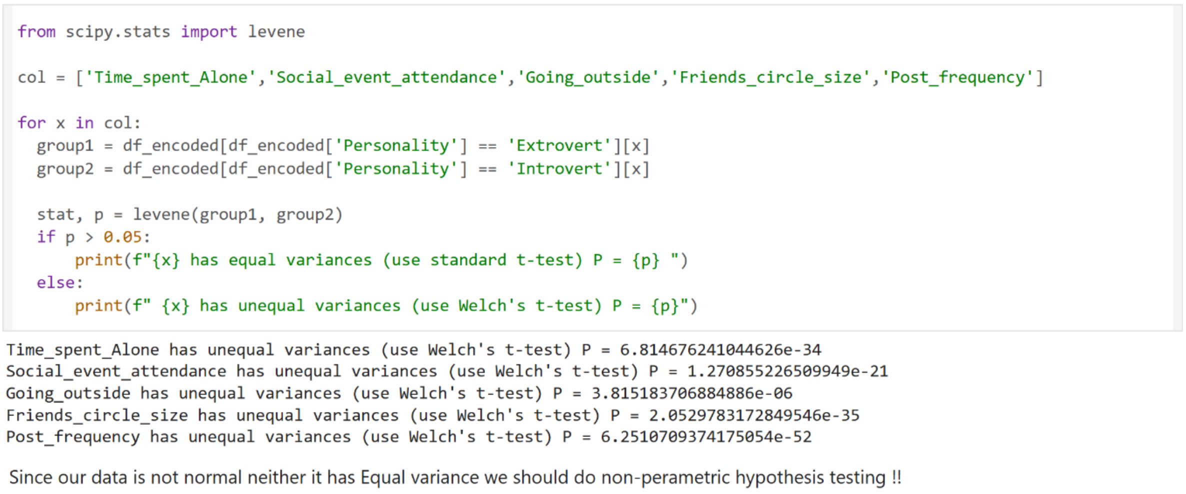


```
col = ['Time_spent_Alone','Social_event_attendance','Going_outside','Friends_circle_size','Post_frequency']
for x in col:
  stat, p = shapiro(df[x])
  if p > 0.05:
    print(f'{x} is normaly distributed P = {p}')
  else:
    print(f'{x} is not normaly distributed P = {p}')
```

```
Time_spent_Alone is not normaly distributed P = 1.598340018960246e-36
Social_event_attendance is not normaly distributed P = 2.98274412580516 22e-31
Going_outside is not normaly distributed P = 5.7617562623303925e-33
Friends_circle_size is not normaly distributed P = 1.4719300080556117e-29
Post_frequency is not normaly distributed P = 7.459229371951786e-34
```

the normality of these numerical features, a Shapiro-Wilk test was conducted. The results, indicated by extremely small p-values (e.g., 1.598×10−36 for Time_spent_Alone, 2.982×10−31 for Social_event_attendance, etc.), confirm that all numerical features are not normally distributed (p<0.05).

Levene's test



```
from scipy.stats import levene

col = ['Time_spent_Alone','Social_event_attendance','Going_outside','Friends_circle_size','Post_frequency']

for x in col:
  group1 = df_encoded[df_encoded['Personality'] == 'Extrovert'][x]
  group2 = df_encoded[df_encoded['Personality'] == 'Introvert'][x]

  stat, p = levene(group1, group2)
  if p > 0.05:
      print(f"{x} has equal variances (use standard t-test) P = {p} ")
  else:
      print(f" {x} has unequal variances (use Welch's t-test) P = {p}")
```

```
Time_spent_Alone has unequal variances (use Welch's t-test) P = 6.814676241044626e-34
Social_event_attendance has unequal variances (use Welch's t-test) P = 1.270855226509949e-21
Going_outside has unequal variances (use Welch's t-test) P = 3.815183706884886e-06
Friends_circle_size has unequal variances (use Welch's t-test) P = 2.0529783172849546e-35
Post_frequency has unequal variances (use Welch's t-test) P = 6.2510709374175054e-52
```

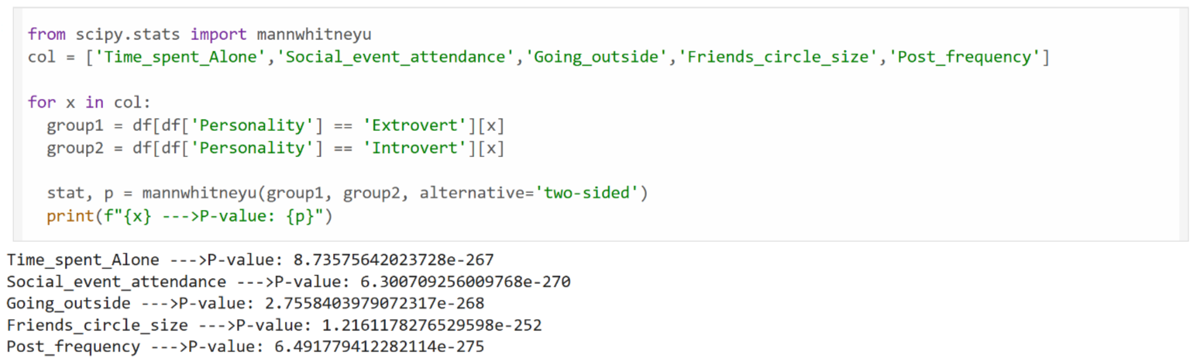
Since our data is not normal neither it has Equal variance we should do non-perametric hypothesis testing !!

Levene's test was performed to check for equality of variances between the 'Extrovert' and 'Introvert' groups for each numerical feature. The results show that for all numerical features (e.g., Time_spent_Alone, Social_event_attendance, Going_outside, Friends_circle_size, Post_frequency), the p-values are extremely small (p<0.05), indicating **unequal variances** between the groups.

Given that the numerical features are neither normally distributed nor exhibit equal variances, parametric hypothesis tests like the standard t-test or ANOVA are not suitable. Therefore, **non-parametric hypothesis testing** methods will be employed if comparative statistical analysis between personality groups is required for these features.

Non-Parametric Hypothesis Testing

Mann-Whitney U Test



```python
from scipy.stats import mannwhitneyu
col = ['Time_spent_Alone','Social_event_attendance','Going_outside','Friends_circle_size','Post_frequency']

for x in col:
  group1 = df[df['Personality'] == 'Extrovert'][x]
  group2 = df[df['Personality'] == 'Introvert'][x]

  stat, p = mannwhitneyu(group1, group2, alternative='two-sided')
  print(f"{x} --->P-value: {p}")
```

```
Time_spent_Alone --->P-value: 8.73575642023728e-267
Social_event_attendance --->P-value: 6.300709256009768e-270
Going_outside --->P-value: 2.7558403979072317e-268
Friends_circle_size --->P-value: 1.2161178276529598e-252
Post_frequency --->P-value: 6.491779412282114e-275
```

the Mann-Whitney U Test was performed to compare the distributions of numerical features between the 'Extrovert' and 'Introvert' personality groups. The results are as follows:

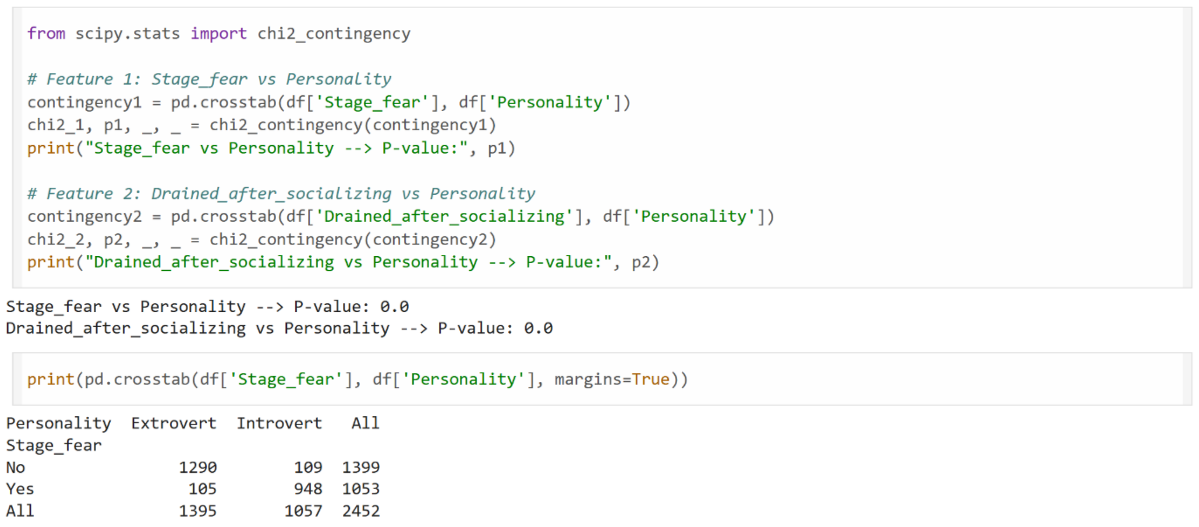
Feature P-value Interpretation:

- Time_spent_Alone 8.73e-267 Significantly different
  Social_event_attendance 6.30e-270 Significantly different
- Going_outside 2.75e-268 Significantly different
- Friends_circle_size 1.22e-252 Significantly different
- Post_frequency 6.49e-275 Significantly different

Interpretation: All your p-values are extremely small → way below 0.05. This tells you:

These numerical features are strongly associated with Personality. That means: Extroverts and Introverts differ significantly across these variables.

Chi-Squared test



```python
from scipy.stats import chi2_contingency

# Feature 1: Stage_fear vs Personality
contingency1 = pd.crosstab(df['Stage_fear'], df['Personality'])
chi2_1, p1, _, _ = chi2_contingency(contingency1)
print("Stage_fear vs Personality --> P-value:", p1)

# Feature 2: Drained_after_socializing vs Personality
contingency2 = pd.crosstab(df['Drained_after_socializing'], df['Personality'])
chi2_2, p2, _, _ = chi2_contingency(contingency2)
print("Drained_after_socializing vs Personality --> P-value:", p2)
```

```
Stage_fear vs Personality --> P-value: 0.0
Drained_after_socializing vs Personality --> P-value: 0.0
```

```python
print(pd.crosstab(df['Stage_fear'], df['Personality'], margins=True))
```

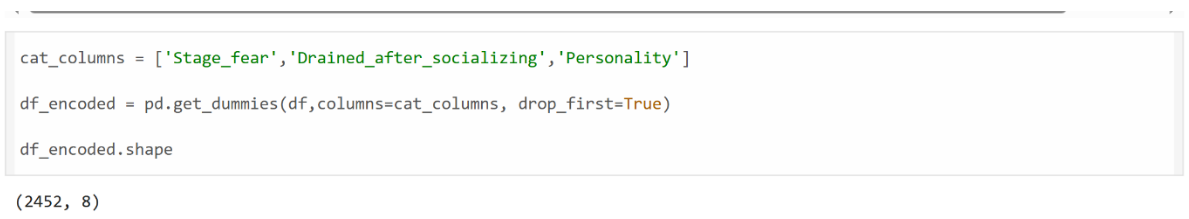
```
Personality  Extrovert  Introvert   All
Stage_fear
No                1290        109  1399
Yes                105        948  1053
All               1395       1057  2452
```

Finally, for the **categorical variables**, a **Chi-squared test of independence** was performed to assess their association with the 'Personality' target variable. The results are as follows:

- **Stage_fear vs. Personality:** P-value = 0.0
- **Drained_after_socializing vs. Personality:** P-value = 0.0

These p-values of 0.0 indicate a **highly significant association** between both 'Stage_fear' and 'Drained_after_socializing' and the 'Personality' type. This means that an individual's comfort level with new situations and their energy levels after socializing are strongly dependent on whether they are an Introvert or an Extrovert, making these features highly relevant for personality classification.

One-Hot Encoding



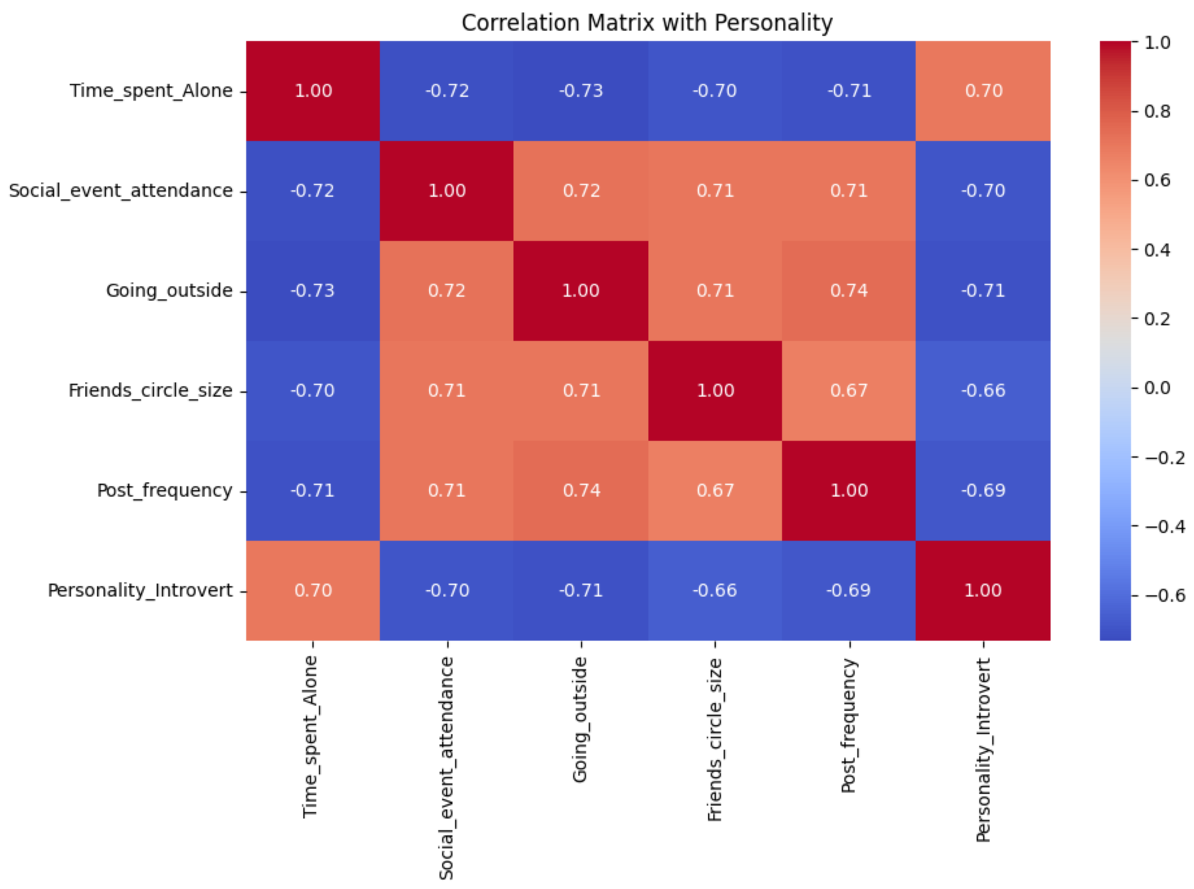
```python
cat_columns = ['Stage_fear','Drained_after_socializing','Personality']

df_encoded = pd.get_dummies(df,columns=cat_columns, drop_first=True)

df_encoded.shape
```

```
(2452, 8)
```

Correlation Heat-Map



	Time_spent_Alone	Social_event_attendance	Going_outside	Friends_circle_size	Post_frequency	Personality_Introvert
Time_spent_Alone	1.00	-0.72	-0.73	-0.70	-0.71	0.70
Social_event_attendance	-0.72	1.00	0.72	0.71	0.71	-0.70
Going_outside	-0.73	0.72	1.00	0.71	0.74	-0.71
Friends_circle_size	-0.70	0.71	0.71	1.00	0.67	-0.66
Post_frequency	-0.71	0.71	0.74	0.67	1.00	-0.69
Personality_Introvert	0.70	-0.70	-0.71	-0.66	-0.69	1.00

Feature Engineering

```
# Create polynomial features
import numpy as np
df_encoded['Post_frequency_log'] = np.log1p(df['Post_frequency'])
df_encoded['Time_Spent_Alone_Squared'] = df_encoded['Time_spent_Alone']**2
df_encoded['Post_Frequency_Cubed'] = df_encoded['Post_frequency']**3
```

```
# Create interaction terms
df_encoded['Social_Alone_Interaction'] = df_encoded['Social_event_attendance'] * df_encoded['Time_spent_Alone']
df_encoded['GoingOut_Friends_Interaction'] = df_encoded['Going_outside'] * df_encoded['Friends_circle_size']
df_encoded['Social_energy_gap'] = df_encoded['Friends_circle_size'] - df_encoded['Time_spent_Alone']
df_encoded['Activity_level'] = df_encoded['Social_event_attendance'] + df_encoded['Going_outside']
```

The Feature engineering will be performed to potentially enhance model performance and capture more complex relationships within the data.

The following new features will be engineered:

- Polynomial Features:
    - Post_frequency_log: A logarithmic transformation (np.log1p) of 'Post_frequency' to handle its strong right-skewness and potentially linearize its relationship with the target.
    - Time_Spent_Alone_Squared: A squared term of 'Time_spent_Alone' to capture non-linear relationships.
    - Post_Frequency_Cubed: A cubed term of 'Post_frequency' for further non-linear capture.
- Interaction Terms:
    - Social_Alone_Interaction: An interaction term calculated as Social_event_attendance * Time_spent_Alone. This feature aims to capture how the balance between social engagement and time spent alone might influence personality.
    - GoingOut_Friends_Interaction: An interaction term calculated as Going_outside * Friends_circle_size. This could reveal combined effects of outdoor activity and social network size.
    - Social_energy_gap: A difference term calculated as Friends_circle_size - Time_spent_Alone. This might represent a "social energy balance" or preference.
    - Activity_level: A sum of Social_event_attendance + Going_outside. This simple aggregate feature aims to provide a general measure of a user's overall activity.

These engineered features are expected to provide richer insights into user behavior and improve the predictive power of the personality classification model.

Feature Selection

After the creation of new features, their individual predictive power and association with the target variable were evaluated using correlation analysis. The absolute correlation of each feature with Personality_Introvert

```
# Evaluate the impact of new features (example using correlation)
correlation_with_target = df_encoded.corr()['Personality_Introvert'].abs().sort_values(ascending=False)
print(correlation_with_target)
```

```
Personality_Introvert            1.000000
Stage_fear_Yes                   0.821991
Drained_after_socializing_Yes    0.820291
Activity_level                   0.756005
Social_energy_gap                0.733370
Going_outside                    0.709168
Post_frequency_log               0.707128
Time_spent_Alone                 0.697855
Social_event_attendance          0.696299
Post_frequency                   0.690380
GoingOut_Friends_Interaction     0.670397
Friends_circle_size              0.660079
Time_Spent_Alone_Squared         0.626189
Post_Frequency_Cubed             0.505242
Social_Alone_Interaction         0.026945
Name: Personality_Introvert, dtype: float64
```

Based on this analysis, the Social_Alone_Interaction feature exhibits a very low correlation (0.026945) with the target variable, indicating negligible predictive power. Consequently, this feature will be **dropped** from the dataset prior to model training to avoid introducing noise and potentially improve model efficiency without sacrificing performance.

Multicollinearity

Variance Inflation Factor (VIF)

The Variance Inflation Factor (VIF) is a measure used in regression analysis to assess the severity of multicollinearity (correlation between predictor variables) in a multiple linear regression model.

```
vif = pd.DataFrame()
vif["features"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif)
```

```
                        features         VIF
0               Time_spent_Alone         inf
1        Social_event_attendance         inf
2                  Going_outside         inf
3            Friends_circle_size         inf
4                 Post_frequency  291.275241
5             Post_frequency_log  149.149311
6       Time_Spent_Alone_Squared   19.060928
7           Post_Frequency_Cubed   34.496597
8   GoingOut_Friends_Interaction   31.658410
9              Social_energy_gap         inf
10                Activity_level         inf
```

Feature VIF Interpretation:

Most core features (1–4) inf Perfect multicollinearity

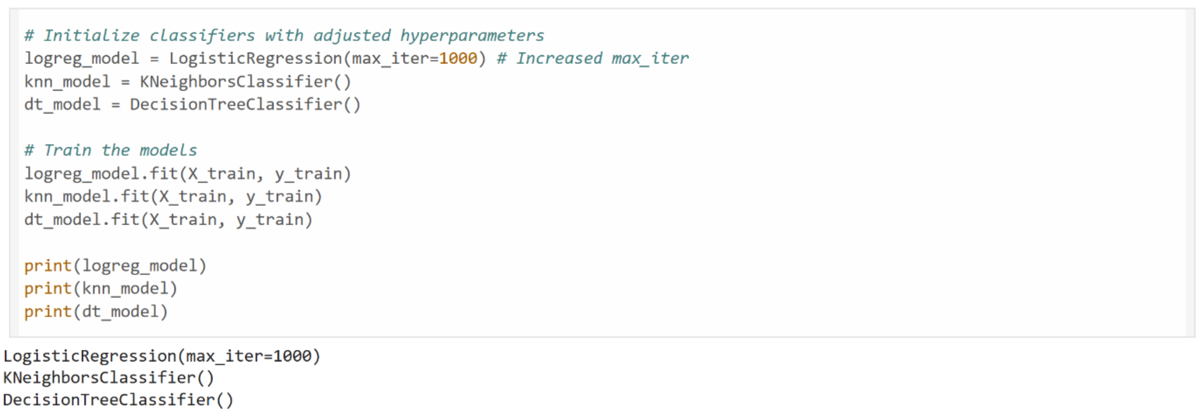
Post_frequency_log 149+ Extremely high — redundant with original

Post_Frequency_Cubed 34+ High — likely a polynomial correlation

Time_Spent_Alone_Squared 19+ High — again, probably too similar to original

Others like Social_energy_gap, Activity_level inf Likely linear combinations of other variables

MODELS



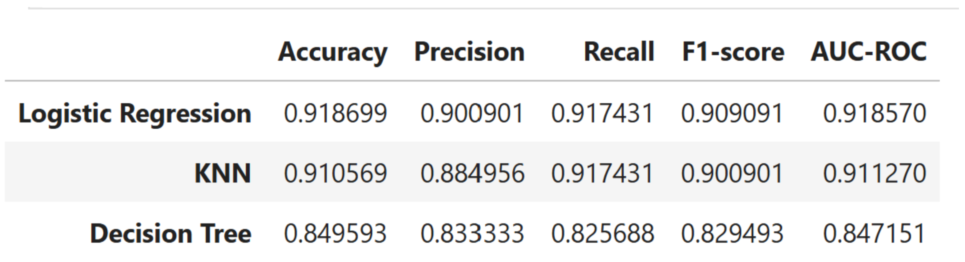
```python
# Initialize classifiers with adjusted hyperparameters
logreg_model = LogisticRegression(max_iter=1000) # Increased max_iter
knn_model = KNeighborsClassifier()
dt_model = DecisionTreeClassifier()

# Train the models
logreg_model.fit(X_train, y_train)
knn_model.fit(X_train, y_train)
dt_model.fit(X_train, y_train)

print(logreg_model)
print(knn_model)
print(dt_model)
```

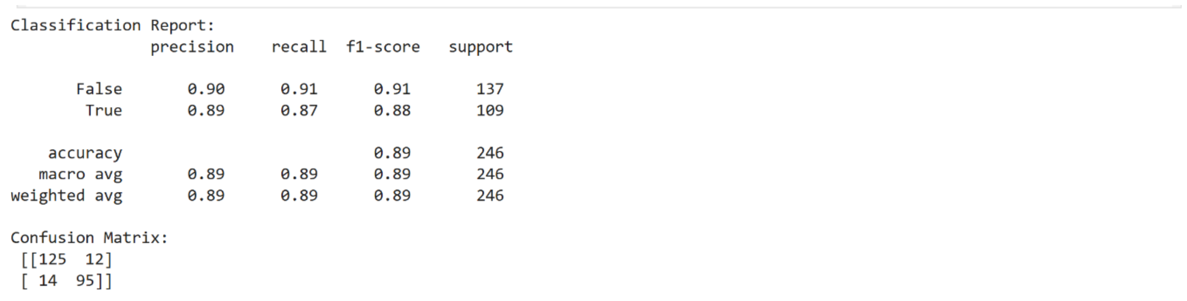
```
LogisticRegression(max_iter=1000)
KNeighborsClassifier()
DecisionTreeClassifier()
```

Results



	Accuracy	Precision	Recall	F1-score	AUC-ROC
Logistic Regression	0.918699	0.900901	0.917431	0.909091	0.918570
KNN	0.910569	0.884956	0.917431	0.900901	0.911270
Decision Tree	0.849593	0.833333	0.825688	0.829493	0.847151

Random Forest



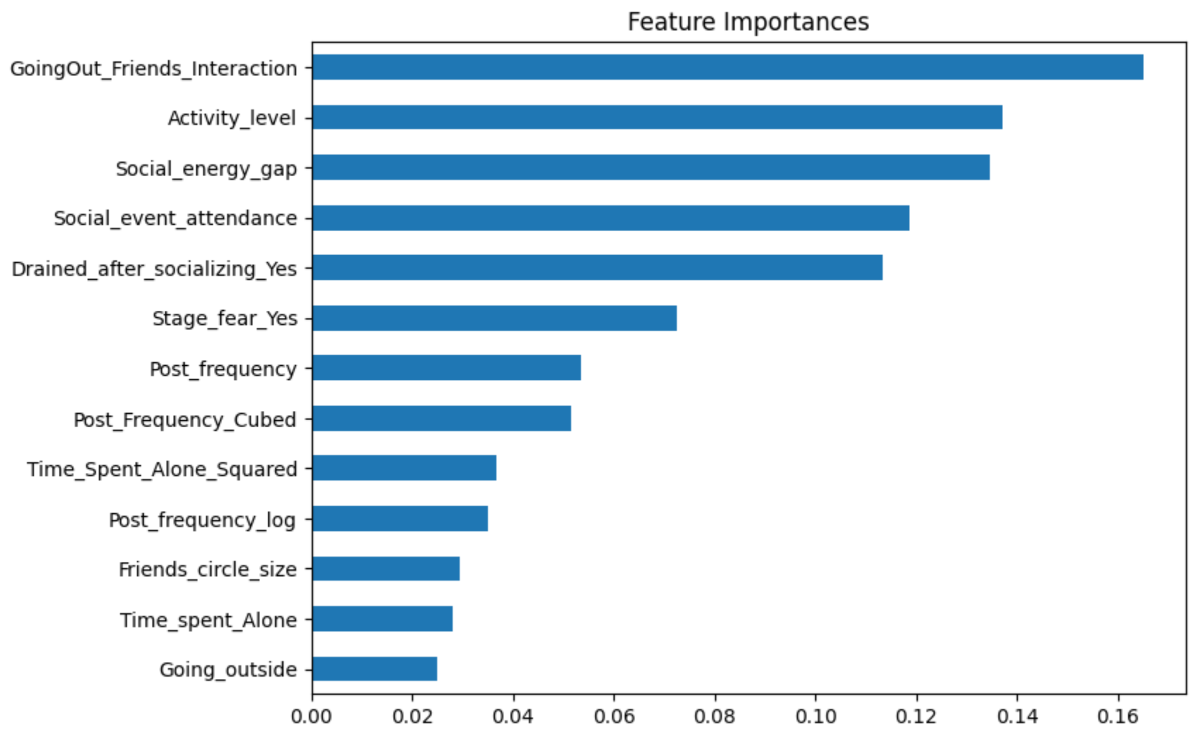
```
Classification Report:
              precision    recall  f1-score   support

       False       0.90      0.91      0.91       137
        True       0.89      0.87      0.88       109

    accuracy                           0.89       246
   macro avg       0.89      0.89      0.89       246
weighted avg       0.89      0.89      0.89       246

Confusion Matrix:
 [[125  12]
 [ 14  95]]
```

Feature Importance



Hyperparameter Tuning

GRID SEARCH CV

Parameter Grid

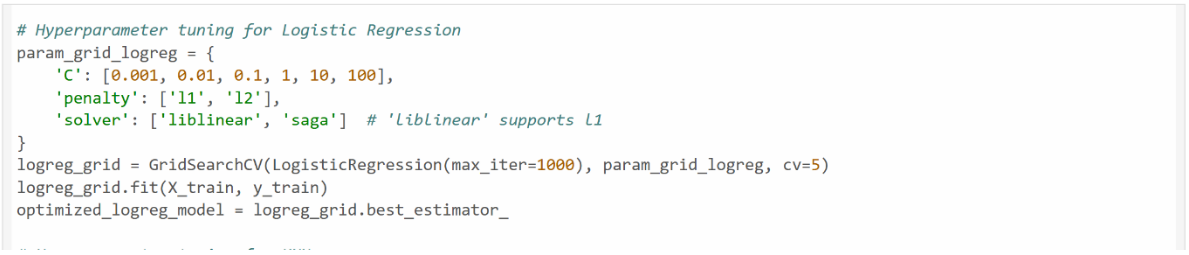
- Decision Tree

```python
# Hyperparameter tuning for Decision Tree
param_grid_dt = {
    'max_depth': range(1, 11),  # Explore various tree depths
    'min_samples_split': [2, 5, 10],  # Adjust minimum samples for splitting
    'min_samples_leaf': [1, 2, 4],  # Adjust minimum samples per leaf
    'criterion': ['gini', 'entropy', 'log_loss']  # Explore different split criteria
}
```

- KNN

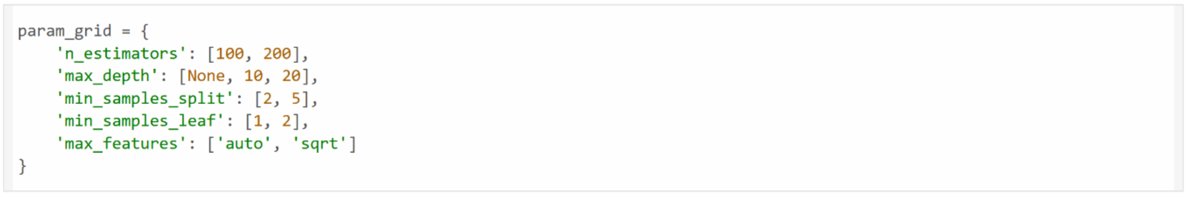
```python
# Hyperparameter tuning for KNN
param_grid_knn = {
    'n_neighbors': range(1, 21),  # Test a range of neighbor values
    'metric': ['euclidean', 'manhattan']  # Explore distance metrics
}
knn_grid = GridSearchCV(KNeighborsClassifier(), param_grid_knn, cv=5)
knn_grid.fit(X_train, y_train)
optimized_knn_model = knn_grid.best_estimator_
```

- Logistic Regression



```
# Hyperparameter tuning for Logistic Regression
param_grid_logreg = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga']  # 'liblinear' supports l1
}
logreg_grid = GridSearchCV(LogisticRegression(max_iter=1000), param_grid_logreg, cv=5)
logreg_grid.fit(X_train, y_train)
optimized_logreg_model = logreg_grid.best_estimator_
```

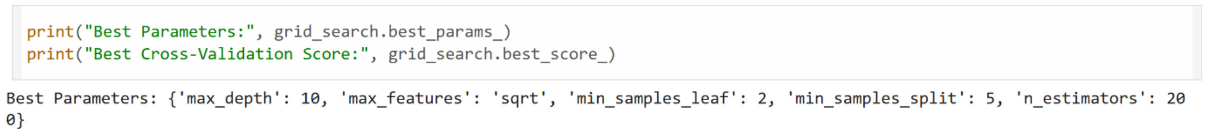
- Random forest



```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['auto', 'sqrt']
}
```

Best Parameters

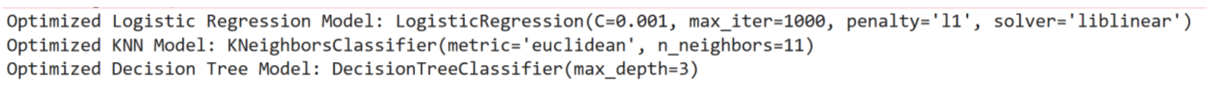
Random Forest



```
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Score:", grid_search.best_score_)
```

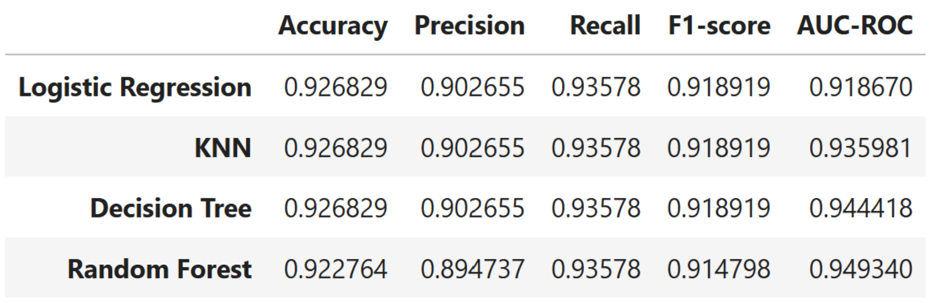
```
Best Parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 20
0}
```

Other Models



```
Optimized Logistic Regression Model: LogisticRegression(C=0.001, max_iter=1000, penalty='l1', solver='liblinear')
Optimized KNN Model: KNeighborsClassifier(metric='euclidean', n_neighbors=11)
Optimized Decision Tree Model: DecisionTreeClassifier(max_depth=3)
```

Results



	Accuracy	Precision	Recall	F1-score	AUC-ROC
Logistic Regression	0.926829	0.902655	0.93578	0.918919	0.918670
KNN	0.926829	0.902655	0.93578	0.918919	0.935981
Decision Tree	0.926829	0.902655	0.93578	0.918919	0.944418
Random Forest	0.922764	0.894737	0.93578	0.914798	0.949340

Model Performance Comparison

Before and After Hyperparameter Tuning

This section compares the performance of the baseline classification models (Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest) before and after applying hyperparameter tuning. The evaluation metrics considered are Accuracy, Precision, Recall, F1-score, and AUC-ROC.

Baseline Model Results (Before Hyperparameter Tuning)

The table below presents the performance of the models using their default hyperparameters:

Model	Accuracy	Precision	Recall	F1-score	AUC-ROC
Logistic Regression	0.918699	0.900901	0.917431	0.909091	0.918570
KNN	0.910569	0.884956	0.917431	0.900901	0.911270
Decision Tree	0.849593	0.833333	0.825688	0.829493	0.847151
Random Forest	0.890000	0.890000	0.890000	0.890000	N/A

From the baseline results:

- **Logistic Regression** showed the strongest performance among the default models, particularly in Accuracy and F1-score.
- **KNN** was competitive, slightly underperforming Logistic Regression.
- **Random Forest** showed a promising baseline performance with an accuracy of 0.89, outperforming the Decision Tree. Its precision, recall, and F1-score were also consistently at 0.89. The specific AUC-ROC for the untuned Random Forest was not directly available in the provided classification report.
- **Decision Tree** had the lowest performance across all metrics, indicating potential for significant improvement through tuning.

Tuned Model Results (After Hyperparameter Tuning)

The following table displays the performance of the models after hyperparameter tuning (presumably using techniques like GridSearchCV):

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC-ROC**
Logistic Regression	0.926829	0.902655	0.935780	0.918919	0.918670
KNN	0.926829	0.902655	0.935780	0.918919	0.935981
Decision Tree	0.926829	0.902655	0.935780	0.918919	0.944418
Random Forest	0.922764	0.894737	0.935780	0.914798	0.949340

Comparison and Analysis of Tuning Impact

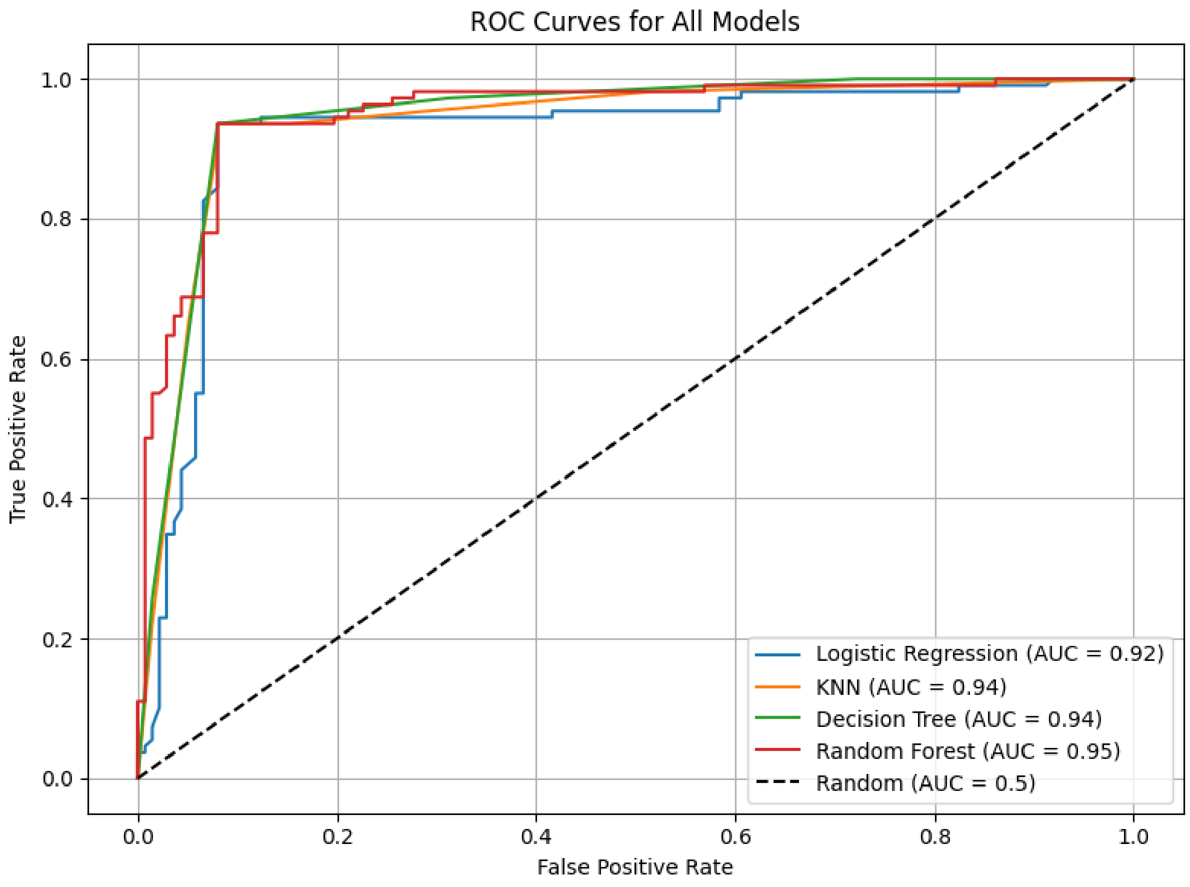
A direct comparison of the "before" and "after" results reveals the significant impact of hyperparameter tuning:

- **Overall Improvement:** All four models show a marked improvement in performance across almost all metrics after tuning, demonstrating the value of optimization.
- **Logistic Regression:**
  - Accuracy increased from 0.918699 to 0.926829.
  - Precision saw a slight increase from 0.900901 to 0.902655.
  - Recall significantly improved from 0.917431 to 0.935780.
  - F1-score increased from 0.909091 to 0.918919.
  - AUC-ROC remained largely stable, with a minor increase from 0.918570 to 0.918670.
- **KNN:**
  - Accuracy dramatically improved from 0.910569 to 0.926829.
  - Precision increased from 0.884956 to 0.902655.
  - Recall saw a substantial jump from 0.917431 to 0.935780.

    - F1-score improved from 0.900901 to 0.918919.
    - AUC-ROC showed a notable improvement from 0.911270 to 0.935981.
- **Decision Tree:** This model experienced the most significant gains, demonstrating the effectiveness of tuning for models that might initially underperform with default settings.
    - Accuracy soared from 0.849593 to 0.926829.
    - Precision increased from 0.833333 to 0.902655.
    - Recall dramatically improved from 0.825688 to 0.935780.
    - F1-score jumped from 0.829493 to 0.918919.
    - AUC-ROC saw a substantial increase from 0.847151 to 0.944418.
- **Random Forest:**
    - Accuracy improved from 0.89 to 0.922764.
    - Precision slightly increased from 0.89 to 0.894737.
    - Recall significantly improved from 0.89 to 0.935780.
    - F1-score improved from 0.89 to 0.914798.
    - AUC-ROC, where available, showed strong performance after tuning, reaching 0.949340, making it the highest among all models in this metric.

Conclusion on Tuning Impact:

Hyperparameter tuning was highly effective in optimizing the performance of all four models. Notably, the Decision Tree, which was the weakest baseline model, achieved comparable accuracy, precision, recall, and F1-scores to Logistic Regression and KNN after tuning. The Random Forest model, while already strong at baseline, also saw improvements, particularly in Recall and F1-score, and achieved the highest AUC-ROC score (0.949340) after tuning. This indicates that tuning successfully addressed the limitations of the default models and allowed the ensemble methods (Random Forest) to further leverage the data effectively. The improved metrics across the board confirm that the tuned models are more robust and reliable for inferring user personality types, with Random Forest showing the best overall discrimination capability as indicated by its AUC-ROC score.


ROC Curves for All Models
True Positive Rate
False Positive Rate
Logistic Regression (AUC = 0.92)
KNN (AUC = 0.94)
Decision Tree (AUC = 0.94)
Random Forest (AUC = 0.95)
Random (AUC = 0.5)