**Evil spirits classifier MLFLOW**

We are going to use MLFLOW to keep track of the various experiments we do on the evil spirits classifier. We are going to check the accuracy of different algorithms and store them on MLFlow for comparison.

In [1]:
```
#getting the packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle

from pickle import dump
from sklearn import metrics
import mlflow.sklearn
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score
from sklearn.compose import ColumnTransformer
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

from sklearn.pipeline import Pipeline
```

In [2]:
```
#importing Mlflow package
import mlflow
import mlflow.sklearn
import os
```

In [3]:
```
#getting the df
spirit = pd.read_excel('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for ML\\Classifier\\evilspirits
spirit.head()
```

Out[3]:

| | id | bone_length | rotting_flesh | hair_length | has_soul | color | type |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.58 | 0.43 | 0.53 | 0.44 | green | Jinnat |
| **1** | 2 | 0.47 | 0.35 | 0.81 | 0.79 | black | Preta |
| **2** | 3 | 0.78 | 0.51 | 0.64 | 0.88 | black | Jinnat |
| **3** | 4 | 0.57 | 0.88 | 0.42 | 0.64 | green | Bhoot |
| **4** | 5 | 0.41 | 0.25 | 0.44 | 0.28 | green | Jinnat |

In [4]:
```
#seeing dataset information
spirit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             900 non-null    int64  
 1   bone_length    900 non-null    float64
 2   rotting_flesh  900 non-null    float64
 3   hair_length    900 non-null    float64
 4   has_soul       900 non-null    float64
 5   color          900 non-null    object 
 6   type           900 non-null    object 
dtypes: float64(4), int64(1), object(2)
memory usage: 49.3+ KB
```

In [5]:
```
#seeing shape of the dataset
print(spirit.shape)
```

```
(900, 7)
```

In [5]:
```
mlflow.autolog()
# Separate the dependent variable from the independent variables

X = spirit.drop('type',axis=1)
y = spirit['type']

# Preprocessing for numerical data
numerical_transformer = Pipeline(steps=[('scaler', StandardScaler())])

# Preprocessing for categorical data
categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

# Define the column transformer to apply the appropriate transformer to each column
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, ['bone_length', 'rotting_flesh', 'hair_length', 'has_soul']),
        ('cat', categorical_transformer, ['color'])
    ])
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


#transforming the X train and X test
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)


# Train a K-Nearest Neighbors (KNN) model
knn_model = KNeighborsClassifier()
knn_model.fit(X_train_transformed, y_train)

# Make predictions on the test set
knn_predictions = knn_model.predict(X_test_transformed)


# Calculate the accuracy of the KNN model
knn_accuracy = accuracy_score(y_test, knn_predictions)

# Log parameters and metrics
mlflow.log_param("model", "K-Nearest Neighbors (KNN)")
mlflow.log_metric("accuracy", knn_accuracy)
```

```
2023/05/14 22:39:22 INFO mlflow.tracking.fluent: Autologging successfully enabled for sklearn.
2023/05/14 22:39:27 INFO mlflow.tracking.fluent: Autologging successfully enabled for statsmodels.
2023/05/14 22:39:28 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID 'd8298f343fe043
ec98db7fe964452ec2', which will track hyperparameters, performance metrics, model artifacts, and lineage inform
ation for the current sklearn workflow
2023/05/14 22:39:28 WARNING mlflow.sklearn: Training metrics will not be recorded because training labels were
not specified. To automatically record training metrics, provide training labels as inputs to the model trainin
g function.
2023/05/14 22:39:28 WARNING mlflow.sklearn: Failed to infer model signature: the trained model does not specify
a `predict` function, which is required in order to infer the signature
2023/05/14 22:39:28 WARNING mlflow.sklearn: Model was missing function: predict. Not logging python_function fl
avor!
2023/05/14 22:39:47 WARNING mlflow.utils.autologging_utils: MLflow autologging encountered a warning: "C:\Users
\sujoydutta\anaconda3\lib\site-packages\_distutils_hack\__init__.py:33: UserWarning: Setuptools is replacing di
stutils."
2023/05/14 22:39:48 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID 'c9b450f881f549
e0bed36fcd76aab2fa', which will track hyperparameters, performance metrics, model artifacts, and lineage inform
ation for the current sklearn workflow
2023/05/14 22:39:48 WARNING mlflow.sklearn: Training metrics will not be recorded because training labels were
not specified. To automatically record training metrics, provide training labels as inputs to the model trainin
g function.
2023/05/14 22:39:48 WARNING mlflow.sklearn: Failed to infer model signature: the trained model does not specify
a `predict` function, which is required in order to infer the signature
2023/05/14 22:39:48 WARNING mlflow.sklearn: Model was missing function: predict. Not logging python_function fl
avor!
2023/05/14 22:39:57 INFO mlflow.utils.autologging_utils: Created MLflow autologging run with ID '8be3bb7711d14f
f191c569991607ca42', which will track hyperparameters, performance metrics, model artifacts, and lineage inform
ation for the current sklearn workflow
```

In [11]:
```
# Enabling automatic MLflow logging for scikit-learn runs
mlflow.sklearn.autolog(max_tuning_runs=None)

with mlflow.start_run(nested=True):

    # Define the hyperparameters grid to search over
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 5, 10],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['sqrt', 'log2']
    }

    # Create a Random Forest classifier and perform a grid search over the hyperparameters
    rf = RandomForestClassifier(random_state=42)
    clf = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, scoring='accuracy', return_train_score=True, verbose=1)
    clf.fit(X_train_transformed, y_train)
    # Make predictions on the test set
    rf_predictions = clf.predict(X_test_transformed)


# Calculate the accuracy of the KNN model
    rf_accuracy = accuracy_score(y_test, rf_predictions)

# Log parameters and metrics
    mlflow.log_param("model", "Random forest")
    mlflow.log_metric("accuracy", rf_accuracy)

# Disabiling autologging
    mlflow.sklearn.autolog(disable=True)
```

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
```

In [ ]: