Ensemble methods – Feature explainabilkity

1) Permutation Importance for a Classification Task

Let's go through an example of estimating PI of features for a classification task in python. We will be using the sklearn library to train our model and we will implement **Algorithm 1** from scratch. Then, we'll plot the results to rank features according to their PI coefficients.

Classification of Iris Flowers:

1. Load the popular Iris dataset. The dataset is an open-source flower classification dataset that consists of three types of flowers i.e. Setosa, Versicolour, and Virginica. The dataset is made up of 50 samples from each of the three types of iris flowers and for each sample, four features are reported: sepal length, sepal width, petal length and petal width.

```
import pandas as pd
from sklearn.datasets import load_iris

# Set seed value
seed_value = 56

data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.DataFrame(data.target, columns=['Iris_type'])
```

2. **Split the data into training and test sets** with the size of the test set being 30% of the dataset.

```
from sklearn.model_selection import train_test_split
data_size_for_testing = 0.3
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=data_size_fo
```

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=data_size_for_testing, random_state=seed_value, shuffle=True,stratify=y)

3. **Standardize features** to improve model training. Target labels are already encoded as integer classes from default.

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
standscaler = StandardScaler()
standscaler.fit(x_train)
train_features = pd.DataFrame(standscaler.transform(x_train), columns=x_train.co
x_train = train_features
y_train = y_train.reset_index().drop("index", axis=1)

test_features = pd.DataFrame(standscaler.transform(x_test), columns=x_test.colum
x_test= test_features
y_test = y_test.reset_index().drop("index", axis=1)
```

4. **Train a simple random forest model.**

```
from sklearn.ensemble import RandomForestClassifier
model_clf = RandomForestClassifier(criterion='gini', random_state=seed_value)
model_clf.fit(x_train,y_train)
```

5. **Implement PI** as a function called *PI_calculate*():.

```
import copy
import numpy as np
# Set numpy pseudo-random generator at a fixed value
np.random.seed(seed_value)
def PI_calculate(model, data, criterion):
    PI_matrix = []
    original_pred = model.predict(data[0])
    original_error = criterion(data[1],original_pred)
    for feature in range(data[0].shape[1]):
        perbutated_data= copy.deepcopy(data[0])
        np.random.shuffle(perbutated_data.iloc[ : , feature])
        perbutated_pred = model.predict(perbutated_data)
        perbutated_error = criterion(data[1], perbutated_pred)
        PI_matrix.append((original_error - perbutated_error))
    return pd.DataFrame(PI_matrix, index=data[0].columns, columns=['Features']).t

from sklearn.metrics import accuracy_score
PI = PI_calculate(model_clf,[x_test,y_test],accuracy_score)
```

return pd.DataFrame(PI_matrix, index=data[0].columns, columns=['Features']).transpose()

6. **Plot the importance of features** to the model performance.

```
import seaborn as sns
import matplotlib.pyplot as plt
p = sns.barplot(x = PI.columns,y=PI.values.flatten())
p.set_xlabel("Features", fontsize = 20)
p.set_ylabel("Increase in model error", fontsize = 20)
p.set_xticklabels(p.get_xticklabels(),rotation = 90)
plt.plot()
```