Ensemble methods – SHAP

- The SHAP library in Python has inbuilt functions to use Shapley values for interpreting machine learning models. It has optimized functions for interpreting tree-based models and a model agnostic explainer function for interpreting any black-box model for which the predictions are known.

- Lundberg and Lee implemented SHAP in the [SHAP](https://github.com/slundberg/shap) Python package. This implementation works for tree-based models in the scikit-learn machine learning library for Python.

- Also they proposed **TreeSHAP**, an efficient estimation approach for tree-based models.

- [SHAP](https://github.com/slundberg/shap) comes with many global interpretation methods based on aggregations of Shapley values. We will demonstrate them in this kernel.

1) Import libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
```

2) Reading data

```
# Load and preview data
df = pd.read_csv('./data/housing.csv')
df.head()
```

3) Summary of data

```
# View summary of data
df.info()
```

4) Missing values treatment, plot and transformation

```
# Plot the distribution of total bedrooms
df['total_bedrooms'].value_counts().plot.bar()
```

```
# Imputing missing values in total_bedrooms by median
df['total_bedrooms'].fillna(df['total_bedrooms'].median(), inplace=True)
```

```
# now check for missing values in total bedrooms
df.isnull().sum()
```

5) Declare feature vector

```
# Declare feature vector and target variable
X = df[['longitude','latitude','housing_median_age','total_rooms',
        'total_bedrooms','population','households','median_income']]
y = df['median_house_value']
```

6) Train-test Split

```
# Split the data into train and test data:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

7)Building the model

```
# Build the model with Random Forest Classifier :
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(max_depth=6, random_state=0, n_estimators=10)
model.fit(X_train, y_train)
```

8) Generate predictions

```
y_pred = model.predict(X_test)
```

9) Import SHAP and visualizing first local explanaitions

```
# import shap library
import shap

# explain the model's predictions using SHAP
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_train)

# visualize the first prediction's explanation
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[0,:], X_train.iloc[0,:])
```

Interpretation

- The above plot shows features each contributing to push the model output from the base value (the average model output over the training dataset we passed) to the model output. Features pushing the prediction higher are shown in red and those pushing the prediction lower are in blue.
- So, `housing_median_age` pushes the prediction higher and `median_income`,`latitude` and `longitude` pushes the prediction lower.
- The base value of the `median_house_value` is 2.063e+5 = 206300.
- The output value is 70189.83 with `housing_median_age=52`, `median_income=1.975`, `latitude=36.73` and `longitude=-119.8`.
- If we take many explanations such as the one shown above, rotate them 90 degrees, and then stack them horizontally, we can see explanations for an entire dataset as shown below.
- The following plot is interactive. Just scroll the mouse and see the different values.

10) SHAP feature importance:

The idea behind SHAP feature importance is simple. Features with large absolute Shapley values are important. Since we want the global importance, we average the absolute Shapley values per feature across the data.
Next, we sort the features by decreasing importance and plot them. The following figure shows the SHAP feature importance for the trained random forest model.

```
shap_values = shap.TreeExplainer(model).shap_values(X_train)
shap.summary_plot(shap_values, X_train, plot_type="bar")
```

- The above plot shows the SHAP feature importance measured as the mean absolute Shapley values.
- The variable `median_income` was the most important feature, changing the predicted `median_house_value` on average by 56000 on x-axis.
- SHAP is based on magnitude of feature attributions. The feature importance plot is useful, but contains no information beyond the importances. For a more informative plot, we will next look at the summary plot.

11) SHAP Summary Plot

The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value.

The color represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction, so we get a sense of the distribution of the Shapley values per feature. The features are ordered according to their importance.

```
shap.summary_plot(shap_values, X_train)
```

The above plot shows the SHAP summary plot. The summary plot combines feature importance with feature effects.

Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction, so we get a sense of the distribution of the Shapley values per feature. The features are ordered according to their importance.

This plot is made of all the dots in the train data. It demonstrates the following information:

- Feature importance: Variables are ranked in descending order.

- Impact: The horizontal location shows whether the effect of that value is associated with a higher or lower prediction.
- Original value: Color shows whether that variable is high (in red) or low (in blue) for that observation.
- Correlation: A high level of the median_income has a high and positive impact on the median_house_value. The "high" comes from the red color, and the "positive" impact is shown on the X-axis.
- Similarly, housing_median_age is positively correlated with the target variable median_house_value.

12) SHAP Dependence Plot

The SHAP Dependence plot shows the marginal effect one or two features have on the predicted outcome of a machine learning model It tells whether the relationship between the target and a feature is linear, monotonic or more complex.

We can create a dependence plot as follows:-

```
shap.dependence_plot('median_income', shap_values, X_train)
```

The function automatically includes another variable that the chosen variable interacts most with. The above plot shows there is an approximately linear and positive trend between median_income and the target variable, and median_income interacts with housing_median_age frequently.

Now, suppose we want to know longitude and the variable that it interacts the most.

We can do

```
shap.dependence_plot('longitude', shap_values, X_train)
```

The plot below shows there exists an approximately linear but negative relationship between longitude and the target variable. This negative relationship is already demonstrated in the variable importance plot. It interacts with median_income variable frequently.