XAI ES

COURSE

LIME ALGORITHM

Explain your model predictions with LIME

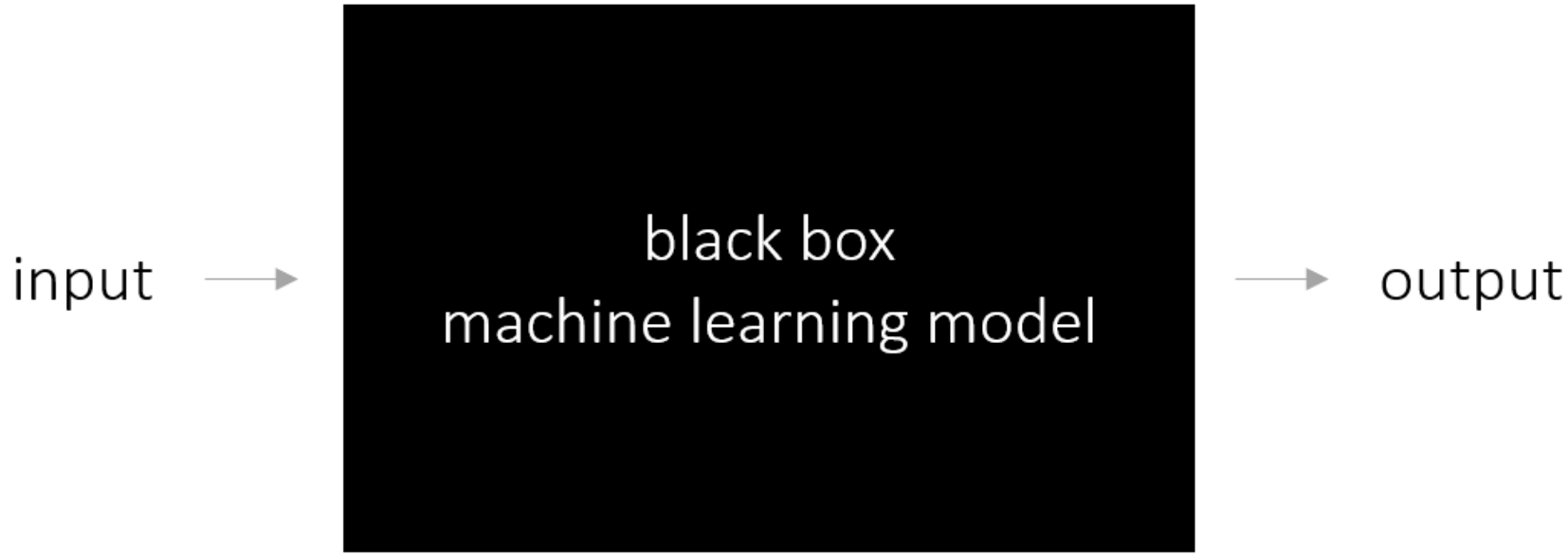
Table of Contents

1. Introduction to LIME

Table of Contents

- LIME (https://christophm.github.io/interpretable-ml-book/lime.html) stands for **Local Interpretable Model-agnostic Explanations**. LIME focuses on training local surrogate models to explain individual predictions. Local surrogate models are interpretable models that are used to explain individual predictions of black box machine learning models. Surrogate models are trained to approximate the predictions of the underlying black box model. Instead of training a global surrogate model, LIME focuses on training local surrogate models.

- LIME is model-agnostic, meaning that it can be applied to any machine learning model. The technique attempts to understand the model by perturbing the input of data samples and understanding how the predictions change.



- Model-specific approaches aim to understand the black model machine learning model by analysing the internal components and how they interact. LIME provides local model interpretability. LIME modifies a single data sample by tweaking the feature values and observes the resulting impact on the output. The most common question is probably: why was this prediction made or which variables caused the prediction.

2. Intuition behind LIME

Table of Contents

- The intuition behind LIME is very simple. First, forget the training data and imagine we have only the black box model where we supply the input data. The black box model generate the predictions for the model. We can enquire the box as many times as we like. Our objective is to understand why the machine learning model made a certain prediction.
- Now, LIME (https://christophm.github.io/interpretable-ml-book/lime.html) comes into play. LIME tests what happens to the predictions when we provide variations in the data which is being fed into the machine learning model.
- LIME (https://christophm.github.io/interpretable-ml-book/lime.html) generates a new dataset consisting of permuted samples and the corresponding predictions of the black box model. On this new dataset LIME then trains an interpretable model (https://christophm.github.io/interpretable-ml-book/simple.html#simple). It is weighted by the proximity of the sampled instances to the instance of interest. The learned model should be a good approximation of the machine learning model predictions locally, but it does not have to be a good global approximation. This kind of accuracy is also called local fidelity.

- Mathematically, local surrogate models with interpretability constraint can be expressed as follows:

```
`explanation(x)=arg ming∈G L(f,g,πx)+Ω(g)`
```

- The explanation model for instance x is the model g (e.g. linear regression model) that minimizes loss function L (e.g. mean squared error). It measures how close the explanation is to the prediction of the original model f (e.g. an xgboost model), while the model complexity Ω(g) is kept low (e.g. prefer fewer features). G is the family of possible explanations.
- In practice, LIME only optimizes the loss part. The user has to determine the complexity, e.g. by selecting the maximum number of features that the linear regression model may use.
- So, the recipe for training local surrogate models is as follows:
  - 1 Select your instance of interest for which you want to have an explanation of its black box prediction.
  - 2 Perturb your dataset and get the black box predictions for these new points.
  - 3 Weight the new samples according to their proximity to the instance of interest.
  - 4 Train a weighted, interpretable model on the dataset with the variations.
  - 5 Explain the prediction by interpreting the local model.

3. Python Implementation of model development

Table of Contents

3.1 Load Preliminaries

Table of Contents

In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input dir
ectory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

In [ ]:
```
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

3.2 Read Data

Table of Contents

In [ ]:
```
# Read and preview data
df = pd.read_csv('/kaggle/input/boston-housing-dataset/HousingData.csv')
df.head()
```

3.3 View Summary of data

Table of Contents

In [ ]:
```
df.info()
```

3.4 Missing values treatment

Table of Contents

In [ ]:
```
df.isnull().sum()
```

- We can see that there are quite a lot of missing values in the dataset. For convinience, I will fill them by the mean of respective columns.

In [ ]:
```
df = df.fillna(df.mean())
```

- Again check for missing values.

In [ ]:
```
df.isnull().sum()
```

- Now, we can see that there are no missing values in the data.

3.5 Feature Vector and Target Variable

Table of Contents

In [ ]:
```
# Declare feature vector and target variable
X = df[['LSTAT','RM','NOX','PTRATIO','DIS','AGE']]
y = df['MEDV']
```

- Here, I have seelcted the following 6 variables as feature vectors for convinience.
  - 1 `LSTAT` - lower status of the population
  - 2 `RM` - average number of rooms per housing
  - 3 `NOX` - nitric oxides concentration (parts per 10 million)
  - 4 `PTRATIO` - pupil-teacher ratio by town
  - 5 `DIS` - weighted distances to five Boston employment centres
  - 6 `AGE` - proportion of owner-occupied units built prior to 1940
- The target variable is `MEDV` which stands for **Median value of owner-occupied homes**.
- The dataset description can be found at -

  https://www.kaggle.com/kyasar/boston-housing (https://www.kaggle.com/kyasar/boston-housing)

3.6 Train-Test Split

Table of Contents

In [ ]:
```
# 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)
```

3.7 Build the Random Forest model

Table of Contents

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

3.8 Generate Predictions

Table of Contents

In [ ]:
```
y_pred = model.predict(X_test)
```

3.9 Evaluate Performance

Table of Contents

In [ ]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)**(0.5)
mse
```

4. Interpret model predictions with LIME

Table of Contents

4.1 Import LIME package

Table of Contents

In [ ]:
```
import lime
import lime.lime_tabular
```

4.2 Create the Explainer

Table of Contents

In [ ]:
```
# LIME has one explainer for all the models
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X_train.columns.values.tolis
t(),
                                                   class_names=['MEDV'], verbose=True, mode='regression')
```

4.3 Use the explainer to explain predictions

Table of Contents

- Here, I will choose 5 instances and use them to explain the predictions.

Select 5th instance

In [ ]:
```
# Choose the 5th instance and use it to predict the results
j = 5
exp = explainer.explain_instance(X_test.values[j], model.predict, num_features=6)
```

In [ ]:
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```

In [ ]:
```
exp.as_list()
```

Interpretation

- The predicted value of the house price is 21.48.
- The variables `LSTAT` and `NOX` have positive influence while `RM`, `DIS`, `AGE` and `PTRATIO` have negative influence on predicted house prices.
- All the values are in thousands of dollars.

Select 10th instance

In [ ]:
```
# Choose the 10th instance and use it to predict the results
j = 10
exp = explainer.explain_instance(X_test.values[j], model.predict, num_features=6)
```

In [ ]:
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```

In [ ]:
```
exp.as_list()
```

Interpretation

- The predicted value of the house price is 9.78.
- All the variables have negative influence on the predicted house prices.
- All the values are in thousands of dollars.

Select 15th instance

In [ ]:
```
# Choose the 15th instance and use it to predict the results
j = 15
exp = explainer.explain_instance(X_test.values[j], model.predict, num_features=6)
```

In [ ]:
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```

In [ ]:
```
exp.as_list()
```

Interpretation

- The predicted value of the house price is 33.48.
- All the variables except `DIS` have positive influence on the predicted house prices.
- All the values are in thousnads of dollars.

Select 20th instance

In [ ]:
```
# Choose the 20th instance and use it to predict the results
j = 20
exp = explainer.explain_instance(X_test.values[j], model.predict, num_features=6)
```

In [ ]:
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```

In [ ]:
```
exp.as_list()
```

Interpretation

- The predicted value of the house price is 23.75.
- The variables `LSTAT` and `NOX` have positive influence while `RM` , `AGE` , `PTRATIO` and `DIS` have negative influence on predicted house prices.
- All the values are in thousands of dollars.

Select 25th instance

In [ ]:
```
# Choose the 25th instance and use it to predict the results
j = 25
exp = explainer.explain_instance(X_test.values[j], model.predict, num_features=6)
```

In [ ]:
```
# Show the predictions
exp.show_in_notebook(show_table=True)
```

In [ ]:
```
exp.as_list()
```

Interpretation

- The predicted value of the house price is 16.67.
- All the variables have negative influence on predicted house prices.
- All the values are in thousands of dollars.

5. References

Table of Contents

The work done in this kernel is based on the following websites -

- 1 https://christophm.github.io/interpretable-ml-book/ (https://christophm.github.io/interpretable-ml-book/)
- 2 https://christophm.github.io/interpretable-ml-book/lime.html (https://christophm.github.io/interpretable-ml-book/lime.html)
- 3 https://blog.dominodatalab.com/shap-lime-python-libraries-part-2-using-shap-lime/ (https://blog.dominodatalab.com/shap-lime-python-libraries-part-2-using-shap-lime/)
- 4 https://www.analyticsvidhya.com/blog/2017/06/building-trust-in-machine-learning-models/ (https://www.analyticsvidhya.com/blog/2017/06/building-trust-in-machine-learning-models/)
- 5 https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b (https://towardsdatascience.com/understanding-model-predictions-with-lime-a582fdff3a3b)
- 6 https://marcotcr.github.io/lime/tutorials/Using%2Blime%2Bfor%2Bregression.html (https://marcotcr.github.io/lime/tutorials/Using%2Blime%2Bfor%2Bregression.html)

Go to Top