6.1HD - Explainable AI

September 4, 2020

GitHub: https://github.com/applecrumble123/SIT799_Human_Aligned_Artificial_Intelligence

Welcome to your assignment this week!

To better understand explainable AI, in this assignment, we will look at the **LIME** framework to explain potential black-box machine learning models in a model-agnostic way. We use a real-world dataset on Census income, also known as the ***Adult dataset*** available in the *UCI* ML Repository where we will predict if the potential income of people is more than $50K/year or not.

For this assignment, we will use:

- XGBoost (XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.). Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.
- Decision Tree which is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

LIME GitHub

After this assignment you will be able to: use the **LIME** framework to explain potential black-box machine learning models in a model-agnostic way.

Let's get started! Run the following cell to install all the packages you will need.

[1]:

```
#!pip install numpy
#!pip install matplotlib
#!pip install lime
#!pip install shap
#!pip install sklearn
#!pip install xgboost
#!pip install graphviz
#!pip install pydot
#!pip install pydotplus
#!pip install seaborn
```

Run the following cell to load the packages you will need.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import gridspec

import seaborn as sns
import lime
from lime.lime_tabular import LimeTabularExplainer
import shap
import itertools

##
from utils import *
##

from sklearn.model_selection import train_test_split
from collections import Counter
import xgboost as xgb
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

import graphviz
from io import StringIO
from sklearn import datasets,tree
import pydot
import pydotplus
from sklearn.tree import export_graphviz
from IPython.display import Image


import warnings
warnings.filterwarnings('ignore')
#plt.style.use('fivethirtyeight')
%matplotlib inline

shap.initjs()
```

```
<IPython.core.display.HTML object>
```

Next, let's load the census dataset. Run the following cell to load the features `X` and the labels `y`.

[3]:
```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

X_raw, y = shap.datasets.adult(display=True)
```

```
X_raw = X_raw.drop(columns=['Capital Gain']) # These two features are␣
 ↪intentionally removed.
X_raw = X_raw.drop(columns=['Capital Loss']) # These two features are␣
 ↪intentionally removed.

labels = np.array([int(label) for label in y])

print('The shape of X_raw is: ',X_raw.shape)
print('The shape of y is: ',labels.shape)
```

```
The shape of X_raw is:  (32561, 10)
The shape of y is:  (32561,)
```

You've loaded:

- `X_raw`: a DataFrame containing 32,561 instances with 12 features.
- `y`: the list of binary labels for the 32,561 examples. If salary of instance i is more than \$50K: $y^{(i)} = 1$ otherwise: $y^{(i)} = 0$.

1 Understanding the Census Income Dataset

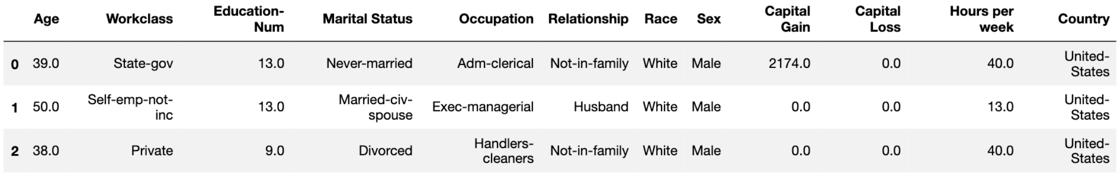
Let's now take a look at our dataset attributes and understand their meaning and significance.

Num	Attribute Name	Type	Description
1	Age	Continuous	Represents age of the person (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked)
2	Workclass	Categorical	Represents the workclass of the person. (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked).
3	Education-Num	Categorical	Numeric representation of educational qualification.Ranges from 1-16.(Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool)
4	Marital Status	Categorical	Represents the marital status of the person(Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse)
5	Occupation	Categorical	Represents the type of profession job of the person(Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces)

Num	Attribute Name	Type	Description
6	Relationship	Categorical	Represents the relationship status of the person(Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried)
7	Race	Categorical	Represents the race of the person(White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black)
8	Sex	Categorical	Represents the gender of the person(Female, Male)
9	Capital Gain	Continuous	The total capital gain for the person
10	Capital Loss	Continuous	The total capital loss for the person
11	Hours per week	Continuous	Total hours spent working per week
12	Country	Categorical	The country where the person was born
13	Income Label	Categorical	The class label column is the one we want to predict

We have a total of 12 features and our objective is to predict if the income of a person will be more than $50K (True) or less than $50K (False). Hence we will be building and interpreting a classification model.

Let's have a look at the first three instances of the dataset (you can use `X.head(3)` to see the content of the dataset):



	Age	Workclass	Education-Num	Marital Status	Occupation	Relationship	Race	Sex	Capital Gain	Capital Loss	Hours per week	Country
0	39.0	State-gov	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States
1	50.0	Self-emp-not-inc	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0	United-States
2	38.0	Private	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	United-States

For example, the first person is 39 years old, works for the state governement, is a Male and was born in the US. By using `y[0:3]` you can get binary values indicating whether these persons have an income higher than $50K or no.

2 Pre-processing

Converting the categorical columns with string values to numeric representations. Typically the XGBoost model can handle categorical data natively being a tree-based model so we don't one-hot encode the features

[4]:

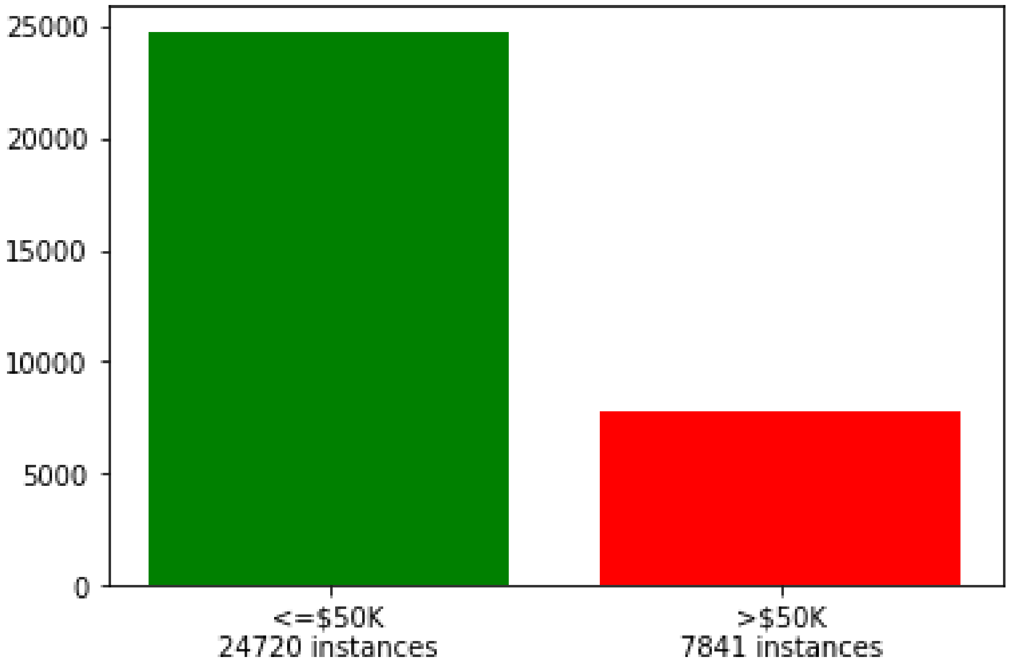
```
mapping = {}
cat_cols = X_raw.select_dtypes(['category']).columns

for col in cat_cols:
    mapping[col] = dict( enumerate(X_raw[col].cat.categories ) )
indices_cat_cols = [ list(X_raw.columns).index(x) for x in cat_cols ]

X = X_raw.copy()
X[cat_cols] = X_raw[cat_cols].apply(lambda x: x.cat.codes)
headers=list(X.columns)
```

```
X.head(3)
```

```
[4]:     Age  Workclass  Education-Num  Marital Status  Occupation  Relationship  \
    0  39.0          7           13.0               4           1             1
    1  50.0          6           13.0               2           4             0
    2  38.0          4            9.0               0           6             1

       Race  Sex  Hours per week  Country
    0     4    1            40.0       39
    1     4    1            13.0       39
    2     4    1            40.0       39
```

Let's have a look at the distribution of people with <= $50K (0) and > $50K (1) income:

```
[5]: plt.bar([0], height=[Counter(labels)[0]], color="green")
     plt.bar([1], height=[Counter(labels)[1]], color="red")
     plt.xticks([0, 1], ['<=$50K\n'+str(Counter(labels)[0])+' instances',
                         '>$50K\n'+str(Counter(labels)[1])+' instances'])
     plt.show()
```



3 Split Train and Test Datasets

As in any Machine Learning, we need to partition the dataset into two subsets – a training and testset. Please note that in practice, the dataset needs to be partitioned into three subsets, the third once being the validation set which will be used for hyperparameters tuning. However, in this assignment, we will not tune the hyperparameters.

Run the following to split the dataset accordingly:

[6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.3,␣
 ↪random_state=42, stratify=y)
print('The shape of training set is: ',X_train.shape)
print('The shape of test set is: ',X_test.shape)
```

```
The shape of training set is:  (22792, 10)
The shape of test set is:  (9769, 10)
```

You've created:

- `X_train`: a trainig DataFrame containing 22,792 instances used for training.
- `y_train`: the list of binary labels for the 22,792 instances of the training set.
- `X_test`: a test DataFrame containing 9,769 instances used for testing.
- `y_test`: the list of binary labels for the 9,769 instances of the test set.

We note that since we are using a stratified splitting, the distribution of samples in the training and test set is similar to the distribution in the dataset, i.e., roughly 24% of positive examples in each subset.

3.1 Training the classification model

Now we train and build a boosting classification model on our training data using XGBoost (XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.). Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Run the following to start the training of the classifier:

[7]:
```
xgc = xgb.XGBClassifier(n_estimators=500, max_depth=5, base_score=0.5,
                        objective='binary:logistic', random_state=42)
xgc.fit(X_train, y_train)
```

[7]:
```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=500, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=42, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
              validate_parameters=False, verbosity=None)
```

4 Predictions on the test data

Now that the classifier is trained, let's make few predictions on the test set:

[8]:

```
predictions = xgc.predict(X_test)
print("The values predicted for the first 20 test examples are:")
print(predictions[:20])

print("The true values are:")
print(y_test[:20])
```

```
The values predicted for the first 20 test examples are:
[1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0]
The true values are:
[1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0]
```

As you can see, our classier is making only 2 errors!

5 Model Performance

Let's now evaluate the performance of our classifier on the test set. For that, we will call `sklearn.metrics.classification_report()` which returns a text report showing the main classification metrics including: Presicion, Recall, and F1-Score. The reported averages include macro average (averaging the unweighted mean per label) and weighted average (averaging the support-weighted mean per label).

[9]:

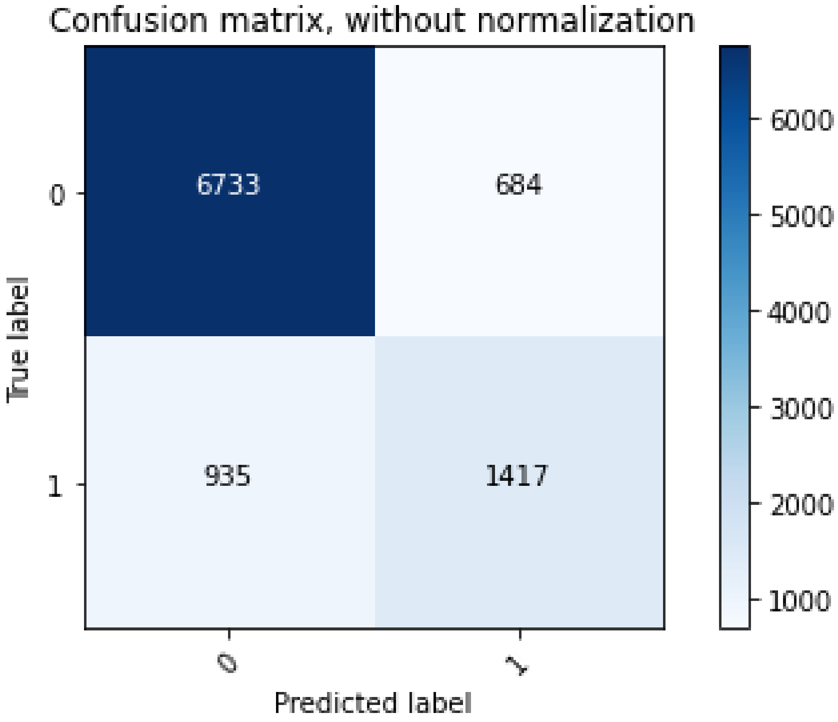
```
report = classification_report(y_test, predictions)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.89      7417
           1       0.67      0.60      0.64      2352

    accuracy                           0.83      9769
   macro avg       0.78      0.76      0.76      9769
weighted avg       0.83      0.83      0.83      9769
```

To get more details, let's print the confusion matrix:

[10]:

```
class_labels = list(set(labels))
cnf_matrix = confusion_matrix(y_test, predictions)
plot_confusion_matrix(cnf_matrix, classes=class_labels,
                      title='Confusion matrix, without normalization')
```



Task 1: Please provide comments on the performance of the classifier.

The classifier has a relatively high accuracy, precision and recall score for label 0 but a relatively low accuracy, precision and recall score for label 1. The classifer is able to perform well to classify label 0 but perform poorly when classifying label 1. The number of correctly classified instances is significantly higher than the wrongly classfied instance for class label 0.

6 Feature importance:

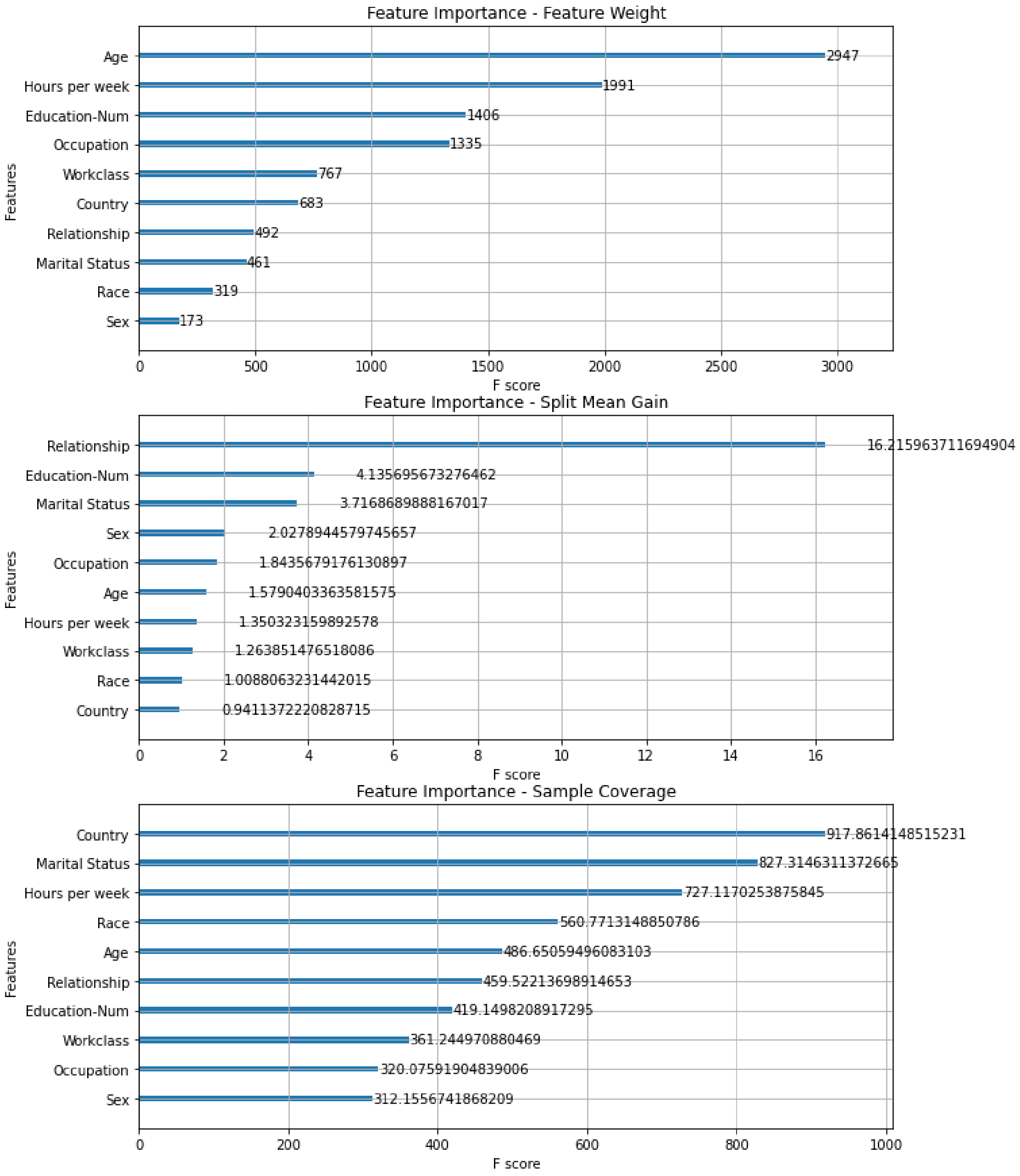
The global feature importance calcuations that come with XGBoost, enables us to view feature importances based on the following:

- **Feature Weights**: This is based on the number of times a feature appears in a tree across the ensemble of trees.
- **Gain**: This is based on the average gain of splits which use the feature.
- **Coverage**: This is based on the average coverage (number of samples affected) of splits which use the feature.

[11]:
```
fig = plt.figure(figsize = (10, 15))
title = fig.suptitle("Default Feature Importances from XGBoost", fontsize=14)
```

```
ax1 = fig.add_subplot(3,1, 1)
xgb.plot_importance(xgc, importance_type='weight', ax=ax1)
t = ax1.set_title("Feature Importance - Feature Weight")

ax2 = fig.add_subplot(3,1, 2)
xgb.plot_importance(xgc, importance_type='gain', ax=ax2)
t = ax2.set_title("Feature Importance - Split Mean Gain")

ax3 = fig.add_subplot(3,1, 3)
xgb.plot_importance(xgc, importance_type='cover', ax=ax3)
t = ax3.set_title("Feature Importance - Sample Coverage")
```

Default Feature Importances from XGBoost



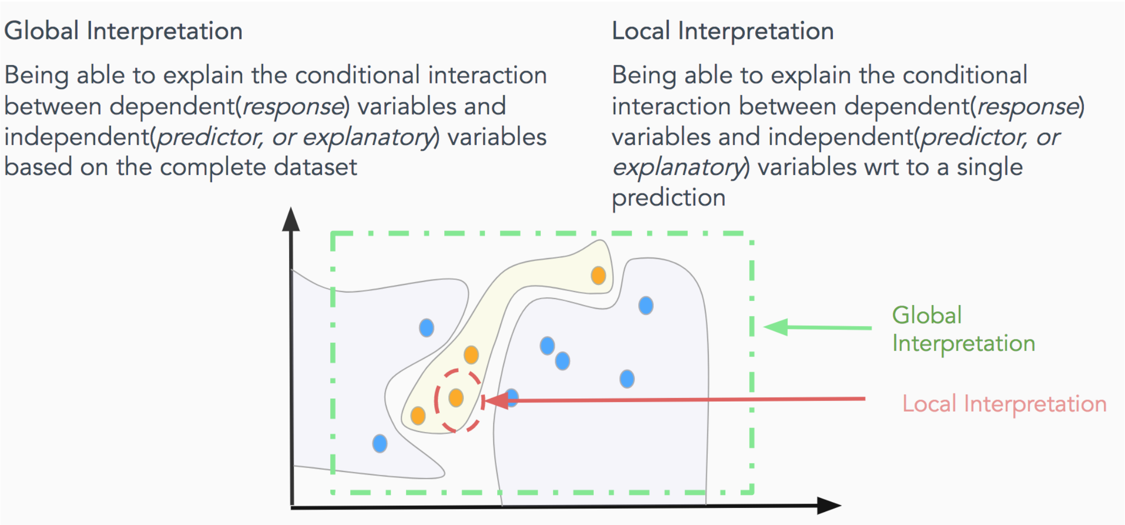
Task 2: Please provide comments on the above global feature importance calcuations.

For the feature weight, it refers to the percentage representing the relative number of times a particular feature occurs in the trees of the model. With respect to the chart above, it shows that "Age" has the highest percentage weight over weights of all features.

For the split mean gain, it is the relative contribution of the corresponding feature to the model calculated by taking each feature's contribution for each tree in the model. The feature with a higher split mean gain when compared to another feature implies it is more important for generating a prediction. With regards to the chart above, it shows that the "Relationship" feature is the most important feature for generating a prediction.

For the sample coverage, it refers to the relative number of observations related to this feature. With respect to the chart above, it shows that "Country" is the feature with the highest observations to decide a leaf node with respect to all the trees in the classifier.

7 Model Interpretation Methods



Global Interpretation

Being able to explain the conditional interaction between dependent(*response*) variables and independent(*predictor, or explanatory*) variables based on the complete dataset

Local Interpretation

Being able to explain the conditional interaction between dependent(*response*) variables and independent(*predictor, or explanatory*) variables wrt to a single prediction

8 LIME:

Lime is able to explain any black box classifier, with two or more classes. All we require is that the classifier implements a function that takes in raw text or a numpy array and outputs a probability for each class. LIME tries to fit a global surrogate model, LIME focuses on fitting local surrogate models to explain why single predictions were made.

Since XGBoost has some issues with feature name ordering when building models with dataframes (we also needed feature names in the previous `plot_importance()` calls), we will build our same model with numpy arrays to make LIME work. Remember the model being built is the same ensemble model which we treat as our black box machine learning model.

Note the difference with the previous `fit` call:

xgc_np.fit(X_train, y_train) vs. xgc_np.fit(X_train.values, y_train)

[12]:

```
xgc_np = xgb.XGBClassifier(n_estimators=500, max_depth=5, base_score=0.5,
                           objective='binary:logistic', random_state=42)
mymodel = xgc_np.fit(X_train.values, y_train)
```

`LimeTabularExplainer` class helps in explaining predictions on tabular (i.e. matrix) data. For numerical features, it perturbs them by sampling from a Normal(0,1) and doing the inverse operation of mean-centering and scaling, according to the means and stds in the training data. For categorical features, it perturbs by sampling according to the training distribution, and making a binary feature that is 1 when the value is the same as the instance being explained.

`explain_instance()` function generates explanations for a prediction. First, we generate neighborhood data by randomly perturbing features from the instance. We then learn locally weighted linear (surrogate) models on this neighborhood data to explain each of the classes in an interpretable way.

[13]:

```
headers=list(X.columns)
explainer = LimeTabularExplainer(X_train.values, feature_names=headers,␣
 ↪discretize_continuous=True,
                                        categorical_features=indices_cat_cols,
                                        class_names=['<= $50K', '> $50K'],verbose=True)
```

8.1 When a person's income <= $50K

Lime shows which features were the most influential in the model taking the correct decision of predicting the person's income as below $50K. The below explanation 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 actual model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in green.

Task 3: Please find a person for which the income is <= $50K and the prediction is correct.

---

[14]:

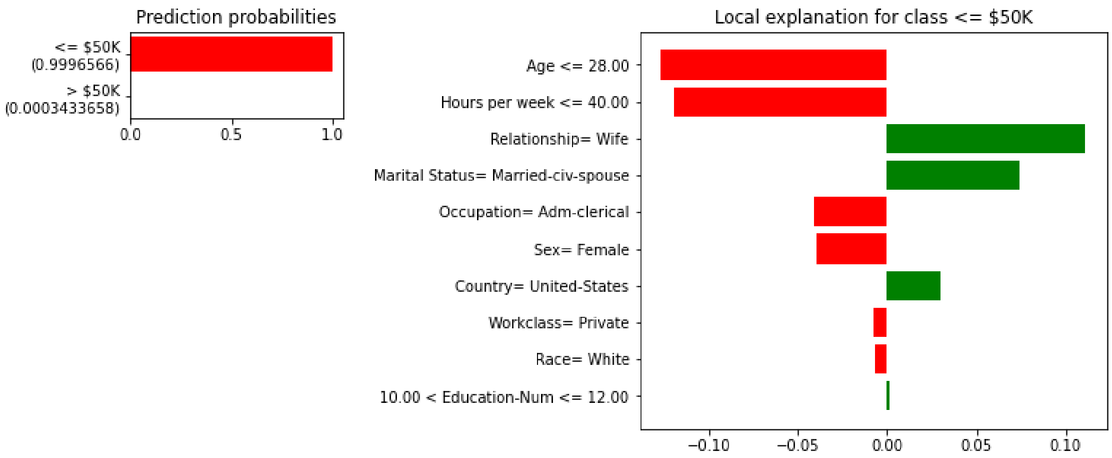
```
# Change only the value of to select that person:
i = 1
###########

exp1 = explainer.explain_instance(X_test.iloc[i].values, xgc_np.predict_proba,
                                  distance_metric='euclidean',
                                  num_features=len(X_test.iloc[i].values))
proba1 = xgc_np.predict_proba(X_test.values)[i]

print("********************")
print('Test id: ' , i)
print('Probability(',exp1.class_names[0],") =", exp1.predict_proba[0])
print('Probability(',exp1.class_names[1],") =", exp1.predict_proba[1])
print('Predicted Label:', predictions[i])
print('True class: ' , y_test[i])
print("********************")
```

```
Intercept 0.211882423531903 67
Prediction_local [0.08707219]
Right: 0.0003433658
********************
Test id:  1
Probability( <= $50K ) = 0.9996566
Probability( > $50K ) = 0.0003433658
Predicted Label: 0
True class:  0
********************
```

The classifier got this example right (it predicted income <= $50K). Let's have a look at the explanation provided by LIME:

[15]:

```
plot_explanation(exp1, mapping)
```



```
<Figure size 432x288 with 0 Axes>
```

Task 4: Please provide comments on the above explanation provided by LIME.

Based on the age, hours per week, sex, occupation, workclass and race, LIME uses these features to explain why the prediction of the instance is in class label 0, which is <= $50K. Based on these features, it then gives a probability of the prediction of class, which is about 99% in this case. The features are categorised in descending order where the first feature contributes the most to the prediction. LIME then shows that these features plays an important role in placing the instance in the predicted class label 0.

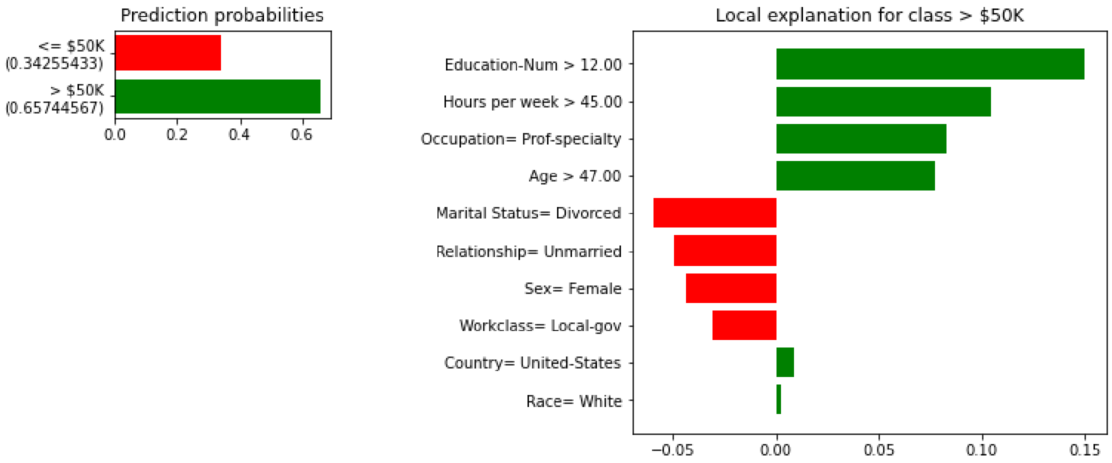
Task 5: Please change the value of one or two features the change the prediction of the classifier:

[16]:
```
instanceModified1 = X_test.iloc[2]
instanceModified1['Age'] = 70
instanceModified1['Hours per week'] = 50
print(instanceModified1)
```

```
Age               70.0
Workclass          2.0
Education-Num     13.0
Marital Status     0.0
Occupation        10.0
Relationship       4.0
Race               4.0
Sex                0.0
Hours per week    50.0
Country           39.0
Name: 3474, dtype: float32
```

[17]:
```
expM1 = explainer.explain_instance(instanceModified1.values, xgc_np.
 ↪predict_proba,
                                   distance_metric='euclidean',
                                  num_features=len(instanceModified1.values))
plot_explanation(expM1, mapping)
```

```
Intercept 0.0632888984456748
Prediction_local [0.30551283]
Right: 0.65744567
```



```
<Figure size 432x288 with 0 Axes>
```

Task 6: How did you choose these features for which you have changed the value? How did you chose these values?

Based on the prediction probability that the classifier predicted the instance to be <=$50K, The “age” and the “Hours per week” are the top 2 features with the highest weightage that contributed to the output of the classifier. Therefore changing these two features will change the prediction of the classifier, classifying the instance in class label 1 instead of class label 0.

8.2 When a person’s income > $50K

Lime shows which features were the most influential in the model taking the correct decision of predicting the person’s income as higher $50K. The below explanation 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 actual model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in green.

Task 7: Please find a person for which the income is > $50K and the prediction is correct.

---

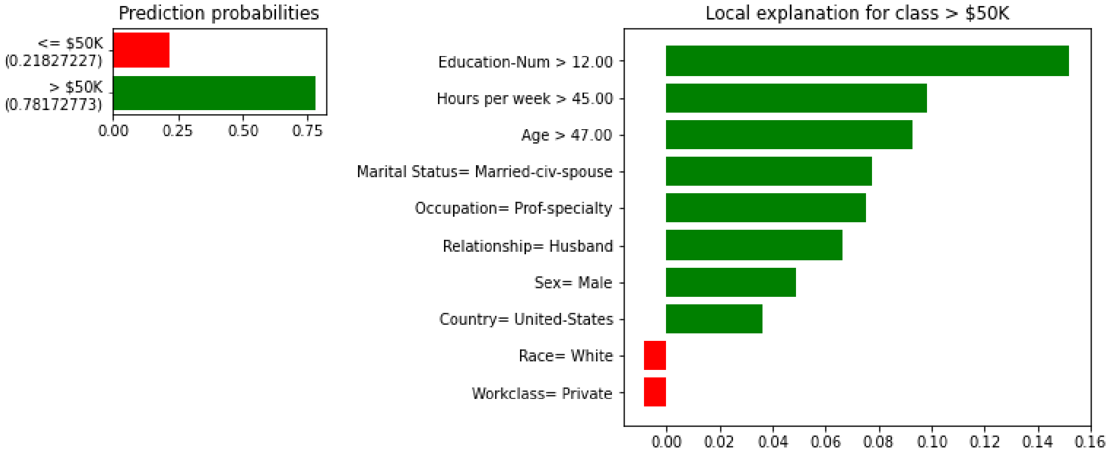
[18]:

```
# Change only the value of to select that person:
j = 60
###########

exp2 = explainer.explain_instance(X_test.iloc[j].values, xgc_np.predict_proba,
                                  distance_metric='euclidean',
                                  num_features=len(X_test.iloc[j].values))
proba2 = xgc_np.predict_proba(X_test.values)[j]

print("********************")
print('Test id: ' , j)
print('Probability(',exp2.class_names[0],") =", exp2.predict_proba[0])
print('Probability(',exp2.class_names[1],") =", exp2.predict_proba[1])
print('Predicted Label:', predictions[j])
print('True class: ' , y_test[j])
print("********************")
```

```
Intercept -0.07542184606838595
Prediction_local [0.55447092]
Right: 0.78172773
********************
Test id:  60
Probability( <= $50K ) = 0.21827227
Probability( > $50K ) = 0.78172773
Predicted Label: 1
True class:  1
********************
```

The classifier got this example right (it predicted income > $50K). Let's have a look at the explanation provided by LIME:

[19]:
```
plot_explanation(exp2, mapping)
```



```
<Figure size 432x288 with 0 Axes>
```

Task 8: Please provide comments on the above explanation provided by LIME.

Based on the Education-Num, hours per week, marital status, occupation, relationship, sex, country and race, LIME uses these features to explain why the prediction of the instance is in class label 1, which is > $50K. Based on these features, it then gives a probability of the prediction of class, which is about 78.1% in this case. The features are categorised in descending order where the first feature contributes the most to the prediction. LIME then shows that these features plays an important role in placing the instance in the predicted class label 1.

Task 9: Please change the value of one or two features the change the prediction of the classifier:

[20]:
```
instanceModified2 = X_test.iloc[j]
instanceModified2['Education-Num'] = 5
instanceModified2['Hours per week'] = 30
print(instanceModified2)
```

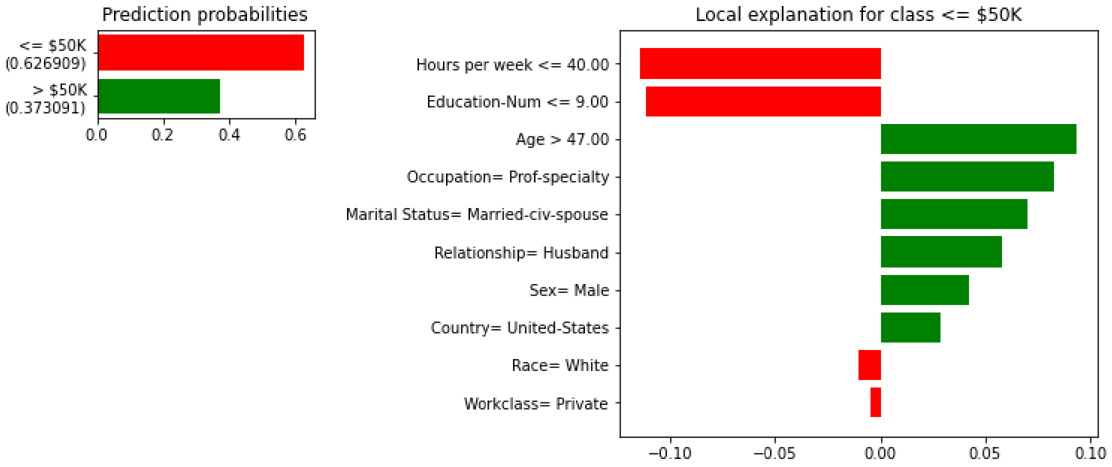
```
Age               57.0
Workclass          4.0
Education-Num      5.0
Marital Status     2.0
Occupation        10.0
```

```
Relationship       0.0
Race               4.0
Sex                1.0
Hours per week    30.0
Country           39.0
Name: 24738, dtype: float32
```

[21]:

```
expM2 = explainer.explain_instance(instanceModified2.values, xgc_np.
 ↪predict_proba,
                                   distance_metric='euclidean',
                                  num_features=len(instanceModified2.values))
plot_explanation(expM2, mapping)
```

```
Intercept 0.13072066070754731
Prediction_local [0.26365301]
Right: 0.373091
```



```
<Figure size 432x288 with 0 Axes>
```

Task 10: How did you choose these features for which you have changed the value? How did you chose these values?

Based on the prediction probability that the classifier predicted the instance to be > $50K, The “Education-Num” and the “Hours per week” are the top 2 features with the highest weightage that contributed to the output of the classifier. Therefore changing these two features will change the prediction of the classifier, classifying the instance in class label 0 instead of class label 1.

9 When a person's income actual is different than predicted

Lime shows which features were the most influential in the model taking the incorrect decision of predicting the person's income. The below explanation 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 actual model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in green.

Task 11: Please find a person for which the the prediction is **incorrect**.

---

[22]:

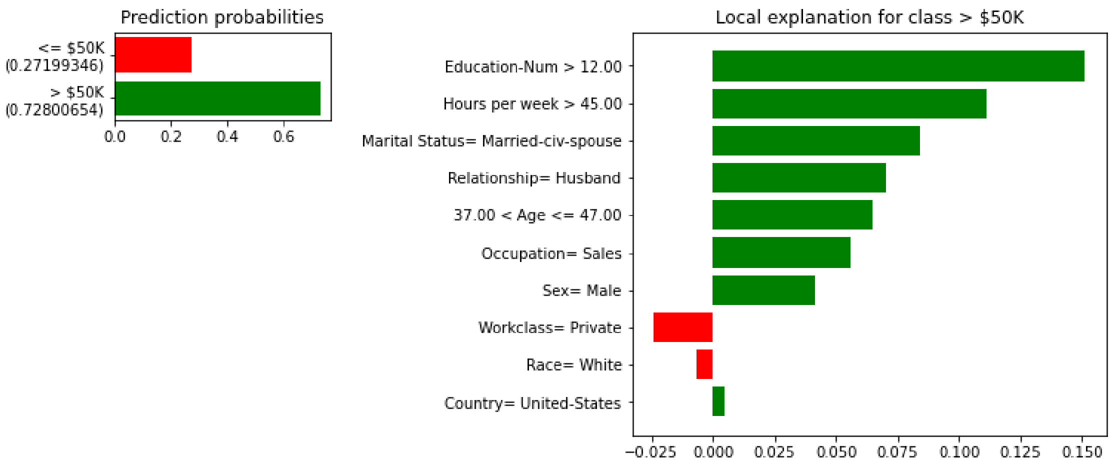
```
# Change only the value of to select that person:
k = 4
###########

exp3 = explainer.explain_instance(X_test.iloc[k].values, xgc_np.predict_proba,
                                  distance_metric='euclidean',
                                  num_features=len(X_test.iloc[k].values))
proba3 = xgc_np.predict_proba(X_test.values)[k]

print("********************")
print('Test id: ' , k)
print('Probability(',exp3.class_names[0],") =", exp3.predict_proba[0])
print('Probability(',exp3.class_names[1],") =", exp3.predict_proba[1])
print('Predicted Label:', predictions[k])
print('True class: ' , y_test[k])
print("********************")
```

```
Intercept -0.03328486161264224
Prediction_local [0.5188057]
Right: 0.72800654
********************
Test id:  4
Probability( <= $50K ) = 0.27199346
Probability( > $50K ) = 0.72800654
Predicted Label: 1
True class:  0
********************
```

The classifier got this example classified incorrectly. Let's have a look at the explanation provided by LIME:

[23]:

```
plot_explanation(exp3, mapping)
```



```
<Figure size 432x288 with 0 Axes>
```

Task 12: Please provide comments on the above explanation provided by LIME.

Based on the Education-Num, hours per week, marital status, age, relationship, sex, occupation and country, LIME uses these features to explain why the prediction of the instance is in class label 1, which is > $50K although the actual class label is 0. Based on these features, it then gives a probability of the prediction of class, which is about 72.18% in this case. The features are categorised in descending order where the first feature contributes the most to the prediction. LIME then shows that these features plays an important role in placing the instance in the predicted class label 1.

Task 13: Please change the value of one or two features the change the prediction of the classifier (to get a correct prediction):

[24]:

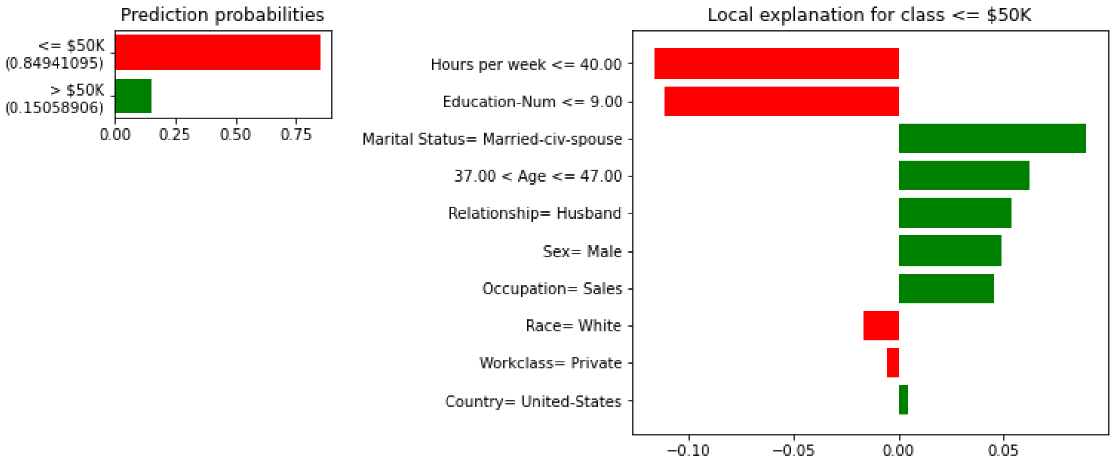
```
instanceModified3 = X_test.iloc[4]
instanceModified3['Education-Num'] = 5
instanceModified3['Hours per week'] = 30
print(instanceModified3)
```

```
Age               42.0
Workclass          4.0
Education-Num      5.0
Marital Status     2.0
Occupation        12.0
Relationship       0.0
Race               4.0
Sex                1.0
```

```
Hours per week    30.0
Country           39.0
Name: 26674, dtype: float32
```

```
[25]: expM3 = explainer.explain_instance(instanceModified3.values, xgc_np.
       ↪predict_proba,
                                          distance_metric='euclidean',
                                         num_features=len(instanceModified3.values))
      plot_explanation(expM3, mapping)
```

```
Intercept 0.15985904178664861
Prediction_local [0.21566091]
Right: 0.15058906
```



```
<Figure size 432x288 with 0 Axes>
```

Task 14: How did you choose these features for which you have changed the value? How did you chose these values?

Based on the prediction probability that the classifier predicted the instance to be > $50K, The “Education-Num” and the “Hours per week” are the top 2 features with the highest weightage that contributed to the output of the classifier. Therefore changing these two features will change the prediction of the classifier, classifying the instance in class label 0 instead of class label 1.

10 Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

Let's use the DecisionTreeClassifier provided by sklearn on our dataset:

```
[26]: tree = DecisionTreeClassifier(random_state=0, max_depth=4).fit(X_train.values,␣
       ↪y_train)
      tree.fit(X_train.values, y_train)
```

```
[26]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                             max_depth=4, max_features=None, max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, presort='deprecated',
                             random_state=0, splitter='best')
```

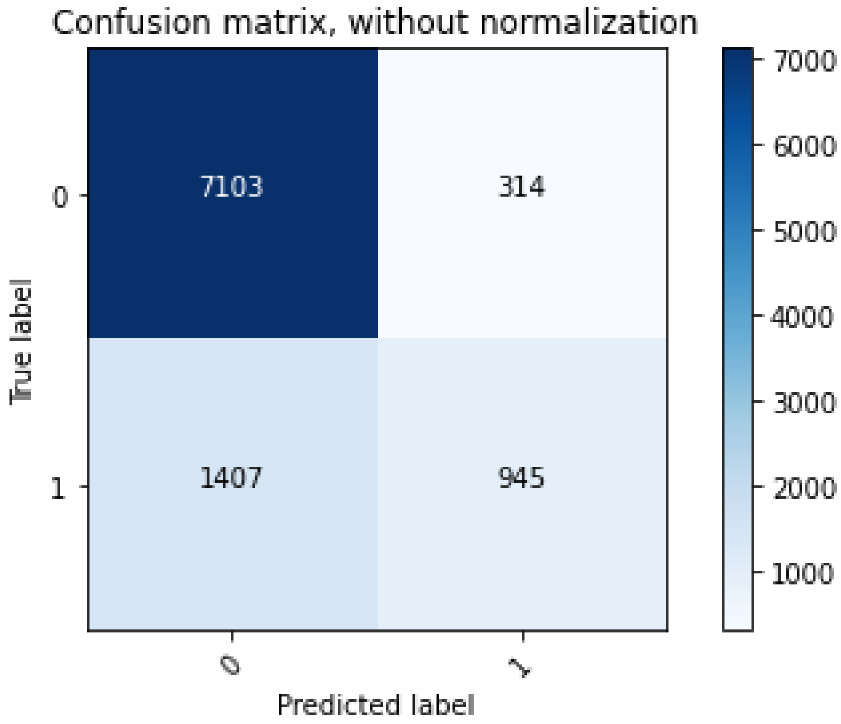
```
[27]: predictions = tree.predict(X_test)
      print("The values predicted for the first 20 test examples are:")
      print(predictions[:20])

      print("The true values are:")
      print(y_test[:20])
```

```
The values predicted for the first 20 test examples are:
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
The true values are:
[1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0]
```

```
[28]: report = classification_report(y_test, predictions)
      print(report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      7417
           1       0.75      0.40      0.52      2352

    accuracy                           0.82      9769
   macro avg       0.79      0.68      0.71      9769
weighted avg       0.81      0.82      0.80      9769
```

```
[29]: class_labels = list(set(labels))
      cnf_matrix = confusion_matrix(y_test, predictions)
      plot_confusion_matrix(cnf_matrix, classes=class_labels,
                            title='Confusion matrix, without normalization')
```



Task 15: Please provide comments on the performance of the decision Tree.

The classifier has a relatively high accuracy, precision and recall score for label 0 but a relatively low accuracy, precision and recall score for label 1. The classifer is able to perform well to classify label 0 but perform poorly when classifying label 1. There is a higher number wrongly classified instances as compared to the correctly classified instance for class label 1.

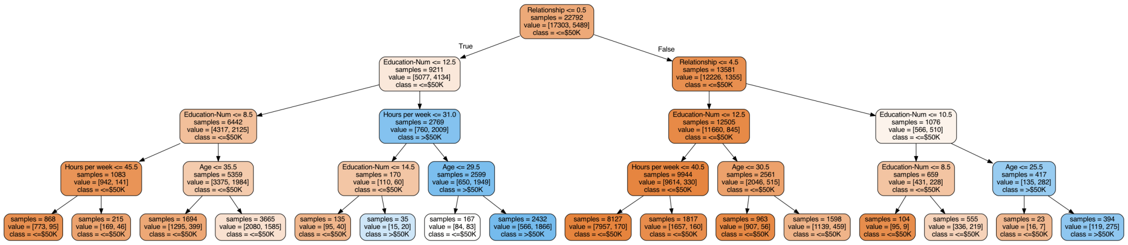
11 Visualzing the Tree

Let's generate a GraphViz representation of the decision tree:

[30]:
```
dot_data = StringIO()
export_graphviz(tree, out_file=dot_data, feature_names=headers,
                filled=True, rounded=True, impurity= False,␣
 ↪class_names=['<=$50K','>$50K'])
```

[31]:
```
graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

[31]:



`Task 16: Explain the tree structure (including the meaning of the colors).`

[32]:
```
"""
At the first layer, which is the root node, the decision tree will predict the number of samples that
belong to either class <=$50k or class >$50k based on the "Relationship" feature where the encoded label is <= 0.5.
Since the most of the samples lie in class <=$50k at 17303 samples at the root node, it shows that the
prediction the root node will make is class <=$50k. If the decision were to end at the root node, it would
predict that all samples belonged to the class <=$50k.

Based on a given feature of a sample, if it is true for Relationship <=0.5, the decision will move to the left
but if it is false for Relationship <=0.5, the decision will move to the right.

On the left cluster, if Relationship <=0.5 is true, the decision tree will make the next decision based on the
Education-Num feature. If the encoded label for Education-Num is of value <= 12.5 is true, the decision will
shift to the left, else, the decision will shift to the right.

On the right cluster, if Relationship <=0.5 is false, the tree will make a decision if the encoded label of the
sample with regards to the Relationship feature is of value <=4.5. If this is true, the decision will shift to
the left. If this is false, the decision will shift to the right.

This process goes on as the tree goes deeper until the nodes are homogeneous and each node will predict the
specific class label if all the samples are predicted to be in that class.

The shade of the nodes represent the predicted target values.
The orange colour represents nodes that have more samples predicted to be class label 0, which is class <= $50k
while the blue colour represents nodes that have more samples predicted to be class label 1, which is class > $50k.

The white colour represents nodes that have almost equal number of samples in each class label.

The lighter the colour, the lower the difference in samples between each class, thus a lower predicted value for
the predicted class. The darker nodes indicate higher predicted values for the predicted class.
"""
```

[32]:

```
'\nAt the first layer, which is the root node, the decision tree will predict
the number of samples that \nbelong to either class <=$50k or class >$50k based
on the "Relationship" feature where the encoded label is <= 0.5. \nSince the
most of the samples lie in class <=$50k at 17303 samples at the root node, it
shows that the \nprediction the root node will make is class <=$50k. If the
decision were to end at the root node, it would \npredict that all samples
belonged to the class <=$50k.\n\nBased on a given feature of a sample, if it is
true for Relationship <=0.5, the decision will move to the left \nbut if it is
false for Relationship <=0.5, the decision will move to the right.\n\nOn the
left cluster, if Relationship <=0.5 is true, the decision tree will make the
next decision based on the \nEducation-Num feature. If the encoded label for
Education-Num is of value <= 12.5 is true, the decision will \nshift to the
left, else, the decision will shift to the right.\n\nOn the right cluster, if
Relationship <=0.5 is false, the tree will make a decision if the encoded label
of the \nsample with regards to the Relationship feature is of value <=4.5. If
this is true, the decision will shift to \nthe left. If this is false, the
decision will shift to the right. \n\nThis process goes on as the tree goes
deeper until the nodes are homogeneous and each node will predict the \nspecific
class label if all the samples are predicted to be in that class.\n\nThe shade
of the nodes represent the predicted target values. \n\nThe orange colour
represents nodes that have more samples predicted to be class label 0, which is
class <= $50k \nwhile the blue colour represents nodes that have more samples
predicted to be class label 1, which is class > $50k.\n\nThe white colour
represents nodes that have almost equal number of samples in each class label.
\n\nThe lighter the colour, the lower the difference in samples between each
class, thus a lower predicted value for \nthe predicted class. The darker nodes
indicate higher predicted values for the predicted class.\n'
```

12 Explanation using LIME

Select any person from the dataset and get the LIME explanation for its classification.

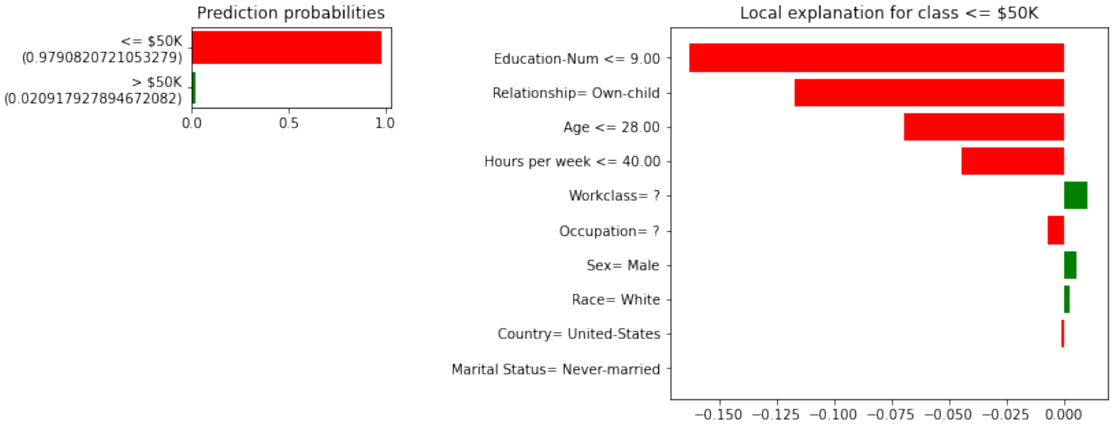
[33]:

```
# Change only the value of to select that person:
h = 5
###########

print(X_test.iloc[h].values.shape)

exp4 = explainer.explain_instance(X_test.iloc[h].values, tree.predict_proba,
                                  distance_metric='euclidean',
                                  num_features=len(X_test.iloc[h].values))
proba1 = tree.predict_proba(X_test.values)[h]

print("********************")
print('Test id: ' , h)
print('Probability(',exp4.class_names[0],") =", exp4.predict_proba[0])
print('Probability(',exp4.class_names[1],") =", exp4.predict_proba[1])
print('Predicted Label:', predictions[h])
print('True class: ' , y_test[h])
print("********************")
```

```
(10,)
Intercept 0.3218318504574811
Prediction_local [-0.06079867]
Right: 0.020917927894672082
********************
Test id:  5
Probability( <= $50K ) = 0.9790820721053279
Probability( > $50K ) = 0.020917927894672082
Predicted Label: 0
True class:  0
********************
```

[34]:

```
plot_explanation(exp4, mapping)
```



```
<Figure size 432x288 with 0 Axes>
```

Task 17: Please provide comments on the above explanation provided by LIME using on the Tree structure above as a context to your explanation.

Based on the Education-Num, relationship, age, hour per week, country, race and sex, LIME uses these features to explain why the prediction of the instance is in class label 0, which is <= $50K. Based on these features, it then gives a probability of the prediction of class, which is about 97.9% in this case. The features are categorised in descending order where the first feature contributes the most to the prediction. LIME then shows that these features plays an important role in placing the instance in the predicted class label 0.

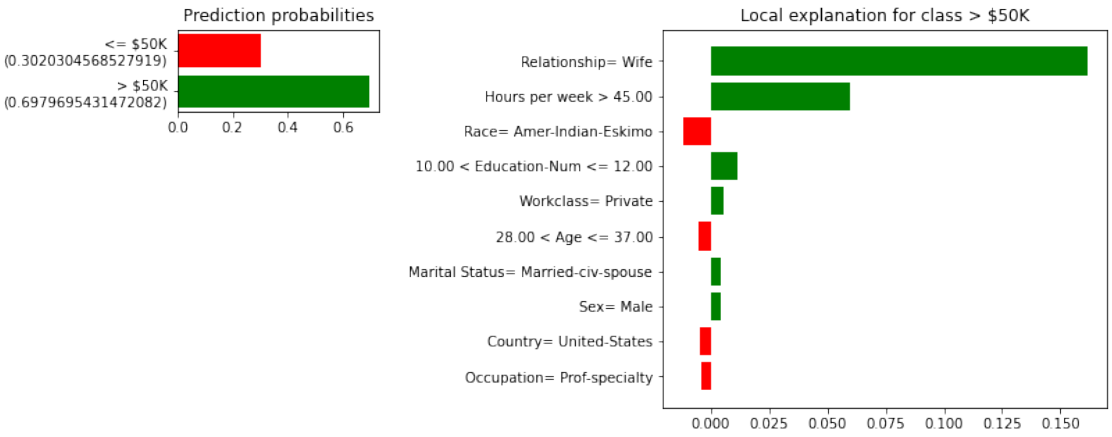
13 Your own test example

Task 18: Following the tree above, create your own test example that will be classified as income > $50K by the decision tree. Explain how you select the values for the features. Use LIME to provide explanation to that test example.

[35]:

```python
sample_instance = np.array([29, 4, 12, 2, 10, 5, 0, 1, 46, 39])

exp4 = explainer.explain_instance(sample_instance, tree.predict_proba,
                                  distance_metric='euclidean',
                                  num_features=len(sample_instance))
proba1 = tree.predict_proba([sample_instance])
print(proba1)

print("********************")
print('Test id: ' , 'sample_instance')
print('Probability(',exp4.class_names[0],") =", exp4.predict_proba[0])
print('Probability(',exp4.class_names[1],") =", exp4.predict_proba[1])
print("********************\n\n")

plot_explanation(exp4, mapping)
```

```
Intercept 0.15996173927825438
Prediction_local [0.38165493]
Right: 0.6979695431472082
[[0.30203046 0.69796954]]
********************
Test id:  sample_instance
Probability( <= $50K ) = 0.3020304568527919
Probability( > $50K ) = 0.6979695431472082
********************
```



```
<Figure size 432x288 with 0 Axes>
```

[36]:
```
"""
Based on the decision tree diagram, if Relationship <= 0.5 is false and␣
 ↪Relationship <= 4.5 is false, it means
that Relationship encoded label can only be 5, Education-Num <= 10.5 is false,␣
 ↪so the encoded label 12 was
selected, age<=25.5 is false, so the value selected is 29. The decision tree␣
 ↪output at the node is predicted to
be class label > $50k. For the rest of the features, it was an estimation using␣
 ↪the XGBoost values where it
predicted provided sample instance as class > $50k as XGBoost is an ensemble␣
 ↪model that uses decision trees.
Therefore, the sameple_instance can be predicted to be class > $50k.

As the decision tree makes the first decision based on the Relationship␣
 ↪feature, this means that the relationship
contributes to the most in making the prediction. Therefore, as seen in the␣
 ↪LIME description, the relationship has
the highest weightage as compared to other features when the classifer decides␣
 ↪to classify the sample_instance
in class label > $50k.

"""
```

[36]:
```
'\nBased on the decision tree diagram, if Relationship <= 0.5 is false and
Relationship <= 4.5 is false, it means \nthat Relationship encoded label can
only be 5, Education-Num <= 10.5 is false, so the encoded label 12 was
\nselected, age<=25.5 is false, so the value selected is 29. The decision tree
output at the node is predicted to\nbe class label > $50k. For the rest of the
features, it was an estimation using the XGBoost values where it \npredicted
```

```
provided sample instance as class > $50k as XGBoost is an ensemble model that
uses decision trees. \nTherefore, the sameple_instance can be predicted to be
class > $50k.\n\nAs the decision tree makes the first decision based on the
Relationship feature, this means that the relationship\ncontributes to the most
in making the prediction. Therefore, as seen in the LIME description, the
relationship has\nthe highest weightage as compared to other features when the
classifer decides to classify the sample_instance \nin class label > $50k.\n\n'
```

14 Congratulations!

You've come to the end of this assignment, and have seen a lot of the ways to explain the predictions given by a classifier.

Congratulations on finishing this notebook!