DSC680 Project 2 Python

April 26, 2020

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as pl
import os, sys
import xgboost
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import shap
```

[3]:
```
# Read the data

df=pd.read_csv('parkinsons.data')
df.head()
```

[3]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	\
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	

	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	…	\
0	0.00007	0.00370	0.00554	0.01109	0.04374	…	
1	0.00008	0.00465	0.00696	0.01394	0.06134	…	
2	0.00009	0.00544	0.00781	0.01633	0.05233	…	
3	0.00009	0.00502	0.00698	0.01505	0.05492	…	
4	0.00011	0.00655	0.00908	0.01966	0.06425	…	

	Shimmer:DDA	NHR	HNR	status	RPDE	DFA	spread1	\
0	0.06545	0.02211	21.033	1	0.414783	0.815285	-4.813031	
1	0.09403	0.01929	19.085	1	0.458359	0.819521	-4.075192	
2	0.08270	0.01309	20.651	1	0.429895	0.825288	-4.443179	
3	0.08771	0.01353	20.644	1	0.434969	0.819235	-4.117501	
4	0.10470	0.01767	19.649	1	0.417356	0.823484	-3.747787	

```
    spread2        D2       PPE
0  0.266482  2.301442  0.284654
1  0.335590  2.486855  0.368674
2  0.311173  2.342259  0.332634
3  0.334147  2.405554  0.368975
4  0.234513  2.332180  0.410335

[5 rows x 24 columns]
```

[15]:
```
#Display statistics for numerical variables
df.describe()
```

[15]:

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	\
count	195.000000	195.000000	195.000000	195.000000	
mean	154.228641	197.104918	116.324631	0.006220	
std	41.390065	91.491548	43.521413	0.004848	
min	88.333000	102.145000	65.476000	0.001680	
25%	117.572000	134.862500	84.291000	0.003460	
50%	148.790000	175.829000	104.315000	0.004940	
75%	182.769000	224.205500	140.018500	0.007365	
max	260.105000	592.030000	239.170000	0.033160	

	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	\
count	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	0.000044	0.003306	0.003446	0.009920	0.029709	
std	0.000035	0.002968	0.002759	0.008903	0.018857	
min	0.000007	0.000680	0.000920	0.002040	0.009540	
25%	0.000020	0.001660	0.001860	0.004985	0.016505	
50%	0.000030	0.002500	0.002690	0.007490	0.022970	
75%	0.000060	0.003835	0.003955	0.011505	0.037885	
max	0.000260	0.021440	0.019580	0.064330	0.119080	

	MDVP:Shimmer(dB)	…	Shimmer:DDA	NHR	HNR	status	\
count	195.000000	…	195.000000	195.000000	195.000000	195.000000	
mean	0.282251	…	0.046993	0.024847	21.885974	0.753846	
std	0.194877	…	0.030459	0.040418	4.425764	0.431878	
min	0.085000	…	0.013640	0.000650	8.441000	0.000000	
25%	0.148500	…	0.024735	0.005925	19.198000	1.000000	
50%	0.221000	…	0.038360	0.011660	22.085000	1.000000	
75%	0.350000	…	0.060795	0.025640	25.075500	1.000000	
max	1.302000	…	0.169420	0.314820	33.047000	1.000000	

	RPDE	DFA	spread1	spread2	D2	PPE
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000
mean	0.498536	0.718099	-5.684397	0.226510	2.381826	0.206552
std	0.103942	0.055336	1.090208	0.083406	0.382799	0.090119
min	0.256570	0.574282	-7.964984	0.006274	1.423287	0.044539

```
25%      0.421306    0.674758   -6.450096    0.174351    2.099125    0.137451
50%      0.495954    0.722254   -5.720868    0.218885    2.361532    0.194052
75%      0.587562    0.761881   -5.046192    0.279234    2.636456    0.252980
max      0.685151    0.825288   -2.434031    0.450493    3.671155    0.527367

[8 rows x 23 columns]
```

[16]:
```
# Search for missing values
print(df.isnull().sum())
```

```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

[17]:
```
# Get column names
column_names = df.columns
print(column_names)
# Get column data types
print(df.dtypes)
```

```
Index(['name', 'MDVP:Fo(Hz)', 'MDVP:Fhi(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)',
       'MDVP:Jitter(Abs)', 'MDVP:RAP', 'MDVP:PPQ', 'Jitter:DDP',
       'MDVP:Shimmer', 'MDVP:Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5',
       'MDVP:APQ', 'Shimmer:DDA', 'NHR', 'HNR', 'status', 'RPDE', 'DFA',
       'spread1', 'spread2', 'D2', 'PPE'],
      dtype='object')
```

```
name                 object
MDVP:Fo(Hz)         float64
MDVP:Fhi(Hz)        float64
MDVP:Flo(Hz)        float64
MDVP:Jitter(%)      float64
MDVP:Jitter(Abs)    float64
MDVP:RAP            float64
MDVP:PPQ            float64
Jitter:DDP          float64
MDVP:Shimmer        float64
MDVP:Shimmer(dB)    float64
Shimmer:APQ3        float64
Shimmer:APQ5        float64
MDVP:APQ            float64
Shimmer:DDA         float64
NHR                 float64
HNR                 float64
status                int64
RPDE                float64
DFA                 float64
spread1             float64
spread2             float64
D2                  float64
PPE                 float64
dtype: object
```

[4]:
```
# Get the features and labels

features=df.loc[:,df.columns!='status'].values[:,1:]
labels=df.loc[:,'status'].values
```

[5]:
```
# Get the count of each label (0 and 1) in labels

print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

[6]:
```
# Scale the features to between -1 and 1

scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

[7]:
```
# Split the dataset

x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2,␣
 ↪random_state=7)
```

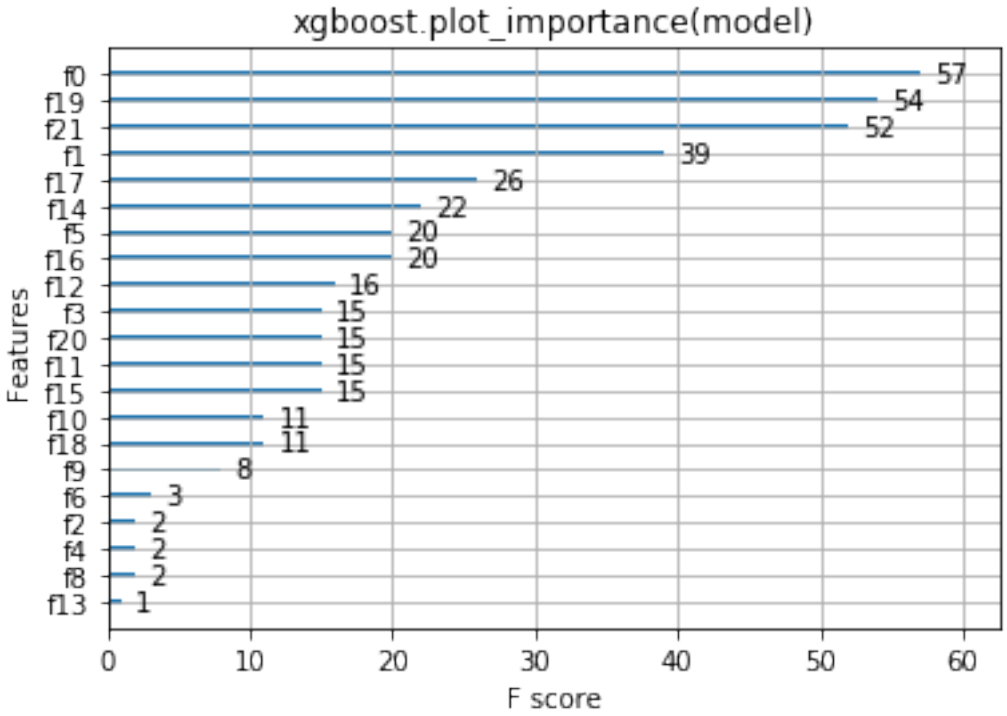
[8]:
```
# Train the model

model=XGBClassifier()
model.fit(x_train,y_train)
```

[8]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

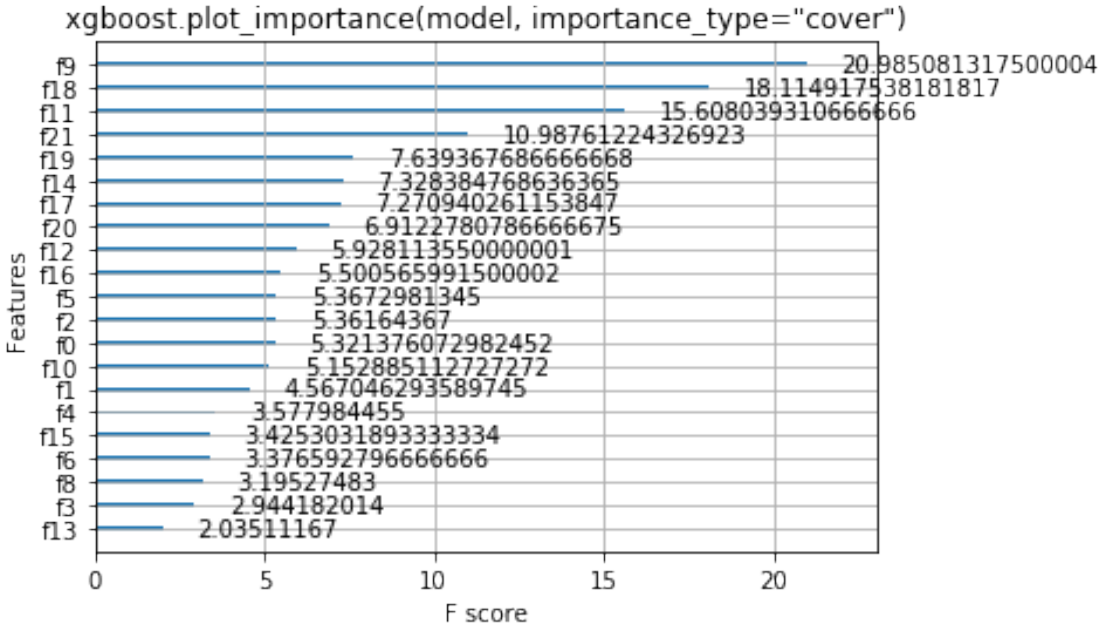
[9]:
```
# Calculate the accuracy

y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
```

```
94.87179487179486
```

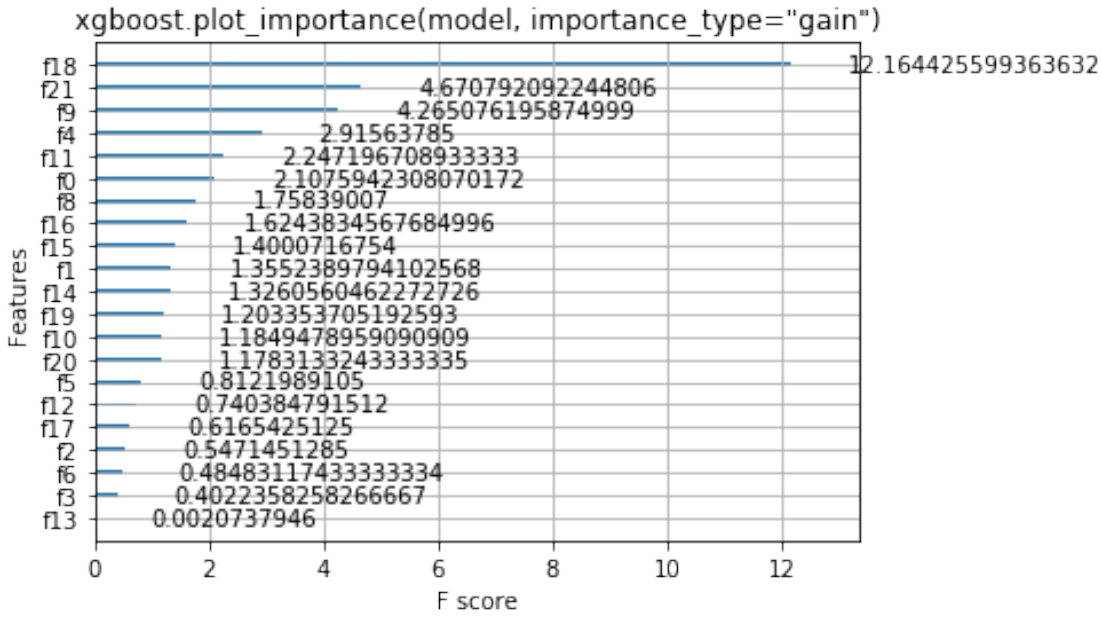
[10]:
```
# Plot the model importance

xgboost.plot_importance(model)
pl.title("xgboost.plot_importance(model)")
pl.show()
```



[11]:
```
xgboost.plot_importance(model, importance_type="cover")
pl.title('xgboost.plot_importance(model, importance_type="cover")')
pl.show()
```

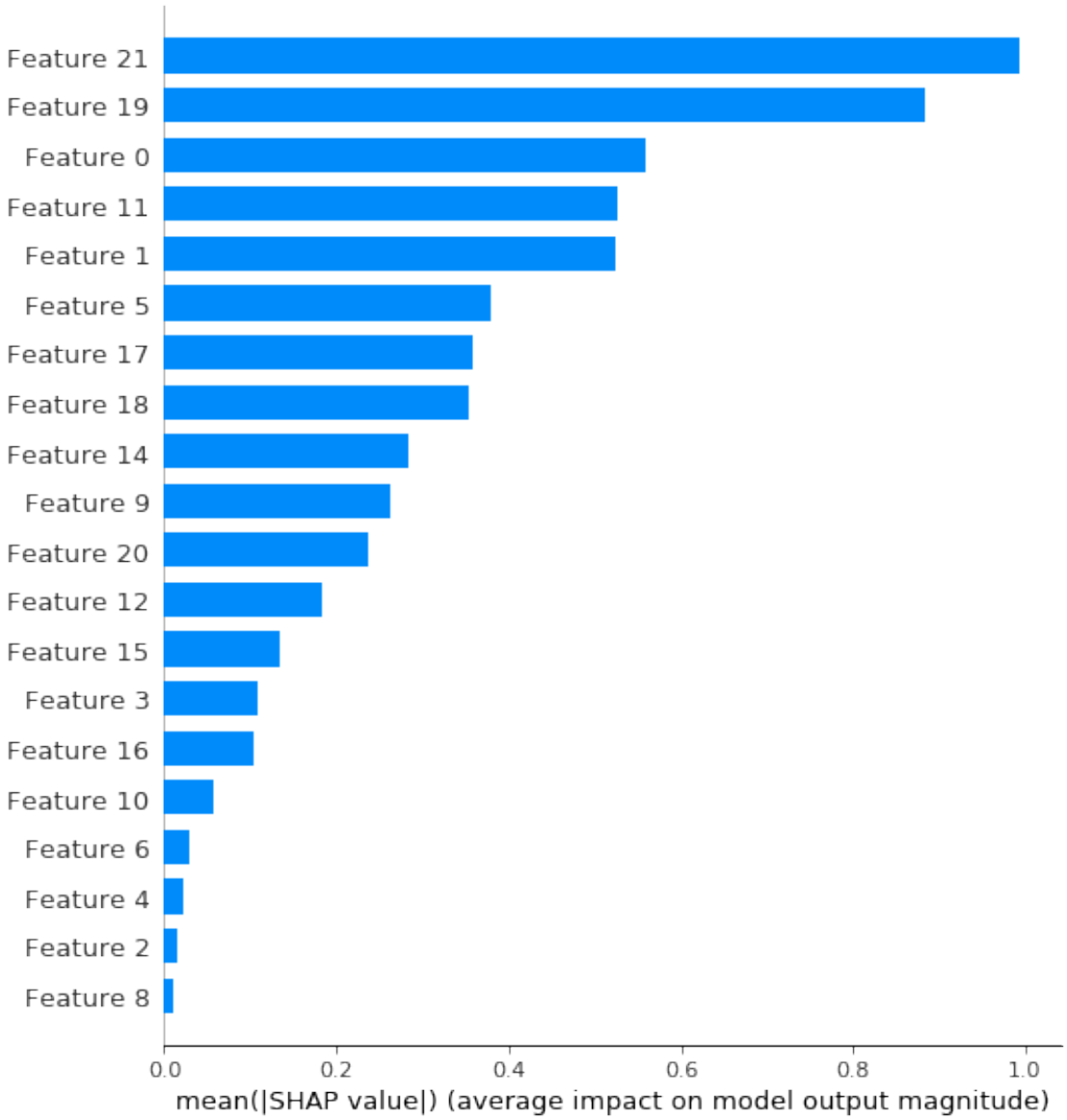


[12]:
```
xgboost.plot_importance(model, importance_type="gain")
pl.title('xgboost.plot_importance(model, importance_type="gain")')
pl.show()
```

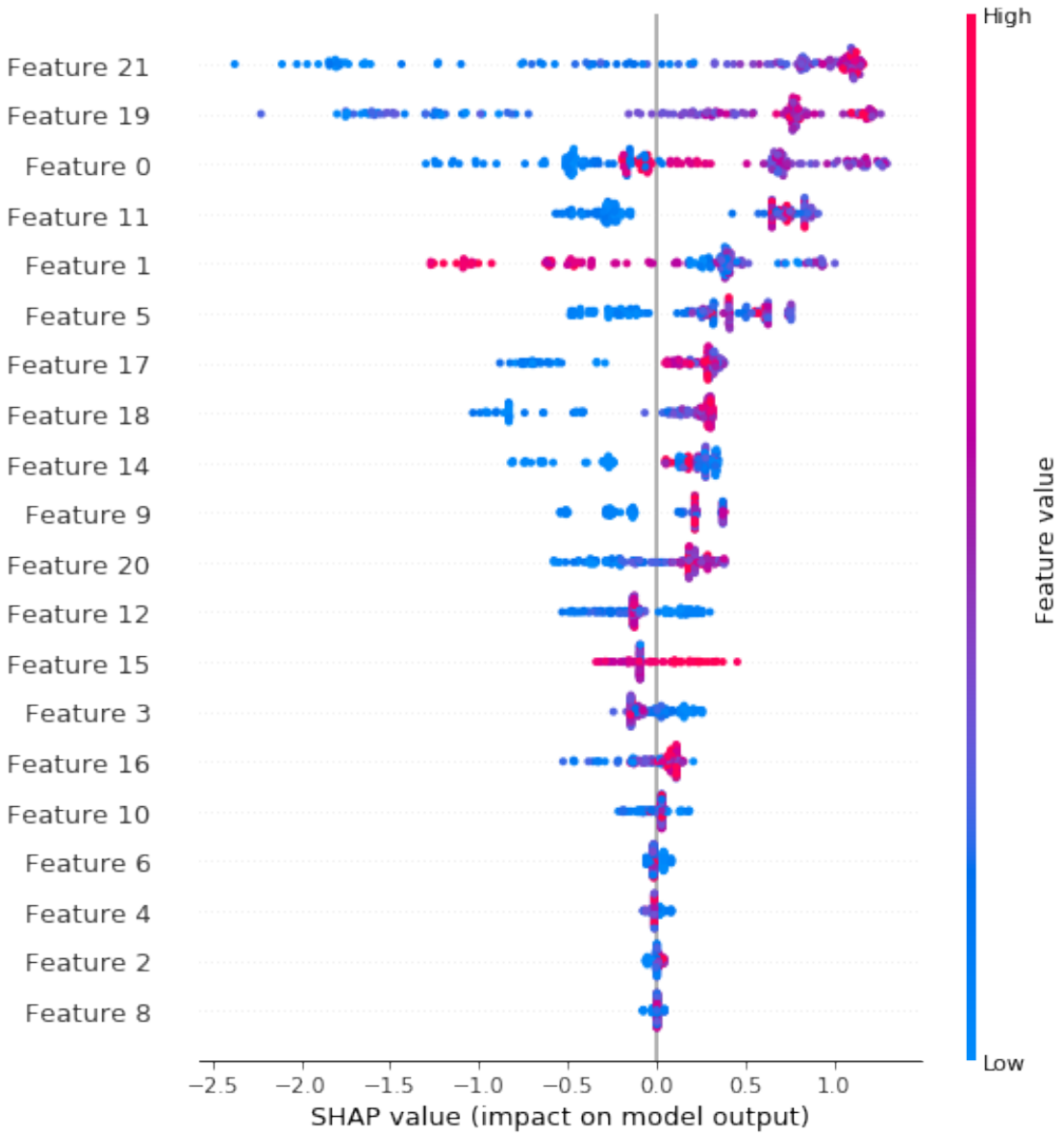


[13]:
```
shap_values = shap.TreeExplainer(model).shap_values(x_train)
shap.summary_plot(shap_values, x_train, plot_type="bar")
```

```
Setting feature_perturbation = "tree_path_dependent" because no background data
was given.
```



[14]:
```
shap.summary_plot(shap_values, x_train)
```


Feature 21
Feature 19
Feature 0
Feature 11
Feature 1
Feature 5
Feature 17
Feature 18
Feature 14
Feature 9
Feature 20
Feature 12
Feature 15
Feature 3
Feature 16
Feature 10
Feature 6
Feature 4
Feature 2
Feature 8
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
High
Feature value
Low

[ ]: