EXP 7 Shap

SIMPLE SHAP on ML MODEL(Local and Global)

[1]:
```
import pandas as pd
import shap
from sklearn.ensemble import RandomForestRegressor
```

[2]:
```
df=pd.read_csv('winequality-red.csv')
df.head()
```

[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

[3]:
```
df.isnull().sum()
```

[3]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

[4]:
```
from sklearn.model_selection import train_test_split
X=df.drop('quality',axis=1)
y=df['quality']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

model=RandomForestRegressor()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

[5]:
```
from sklearn.metrics import r2_score,mean_squared_error
r2=r2_score(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
print("r2:",r2)
print("mse:",mse)
```

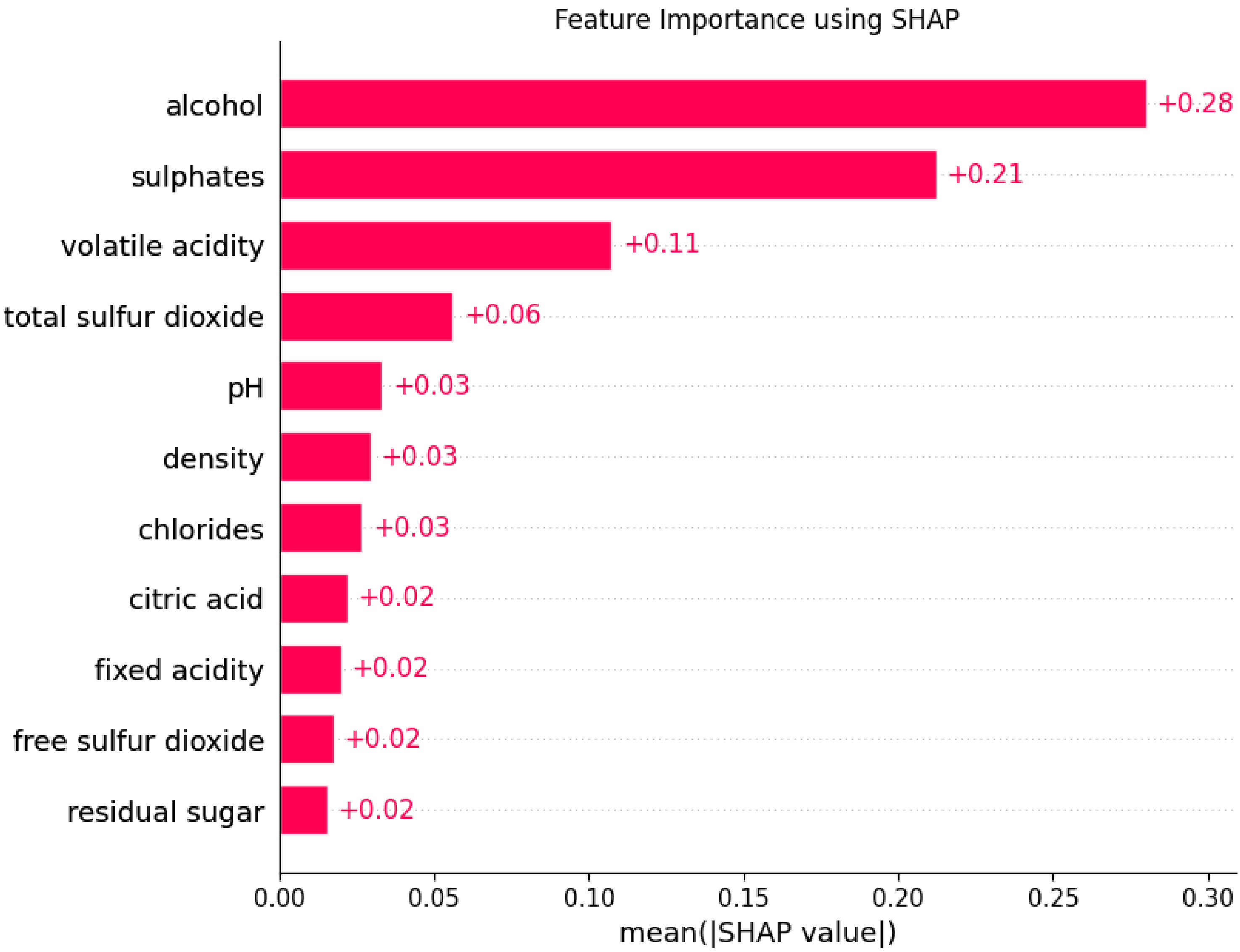
[5]:
```
r2: 0.5210324123193713
mse: 0.313008124999999997
```

[6]:
```
import shap
exp=shap.Explainer(model)
shap_values=exp(X_test)
```

[7]:
```
#Global Explanation

import matplotlib.pyplot as plt
plt.title('Feature Importance using SHAP')
shap.plots.bar(shap_values,max_display=12)
#shap.summary_plot(shap_values, X_test, plot_type="bar", show=False) can use this also...
```

[7]:

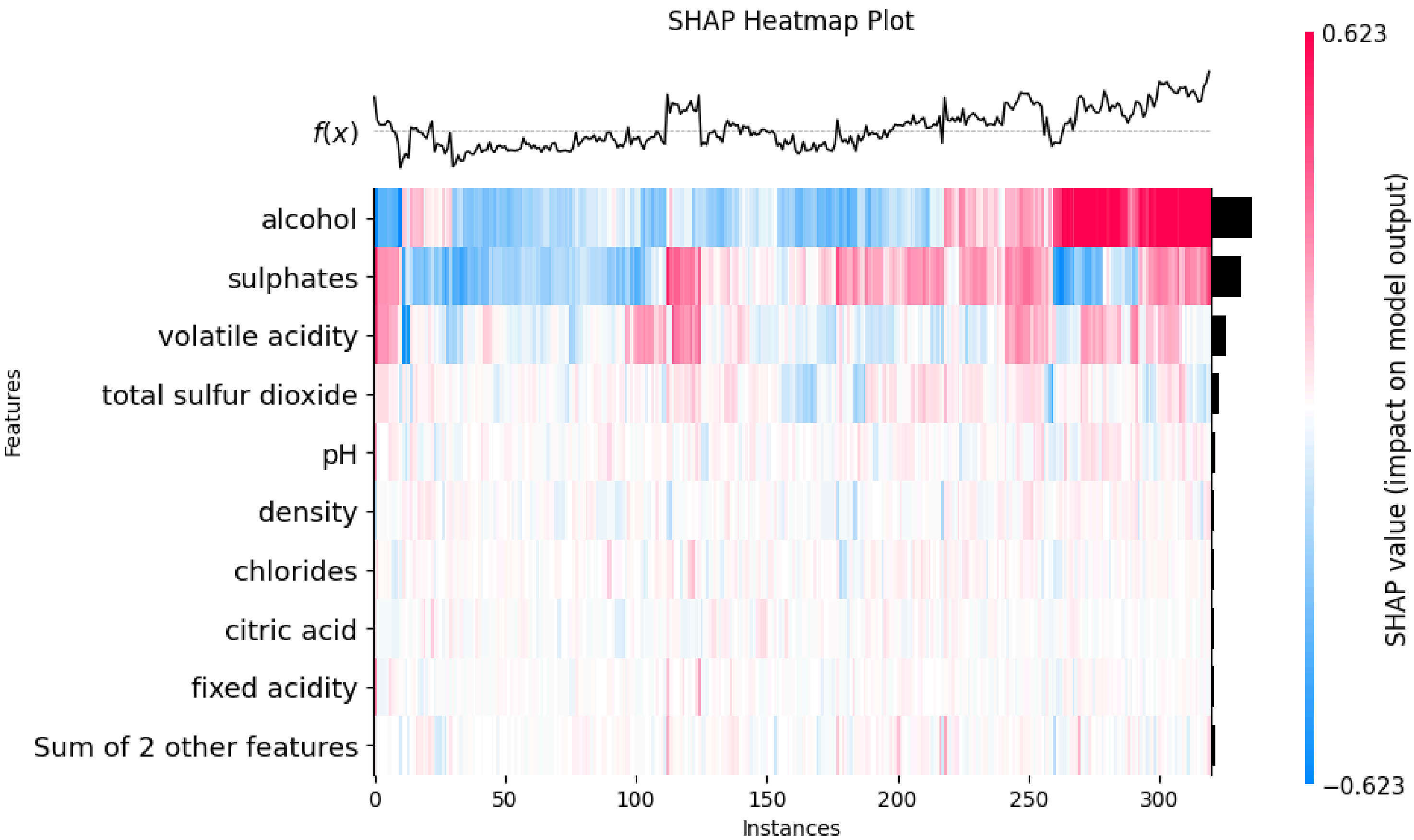


[8]:

```
#Global Explanation

plt.ylabel('Features')
plt.title('SHAP Heatmap Plot')
shap.plots.heatmap(shap_values)
```

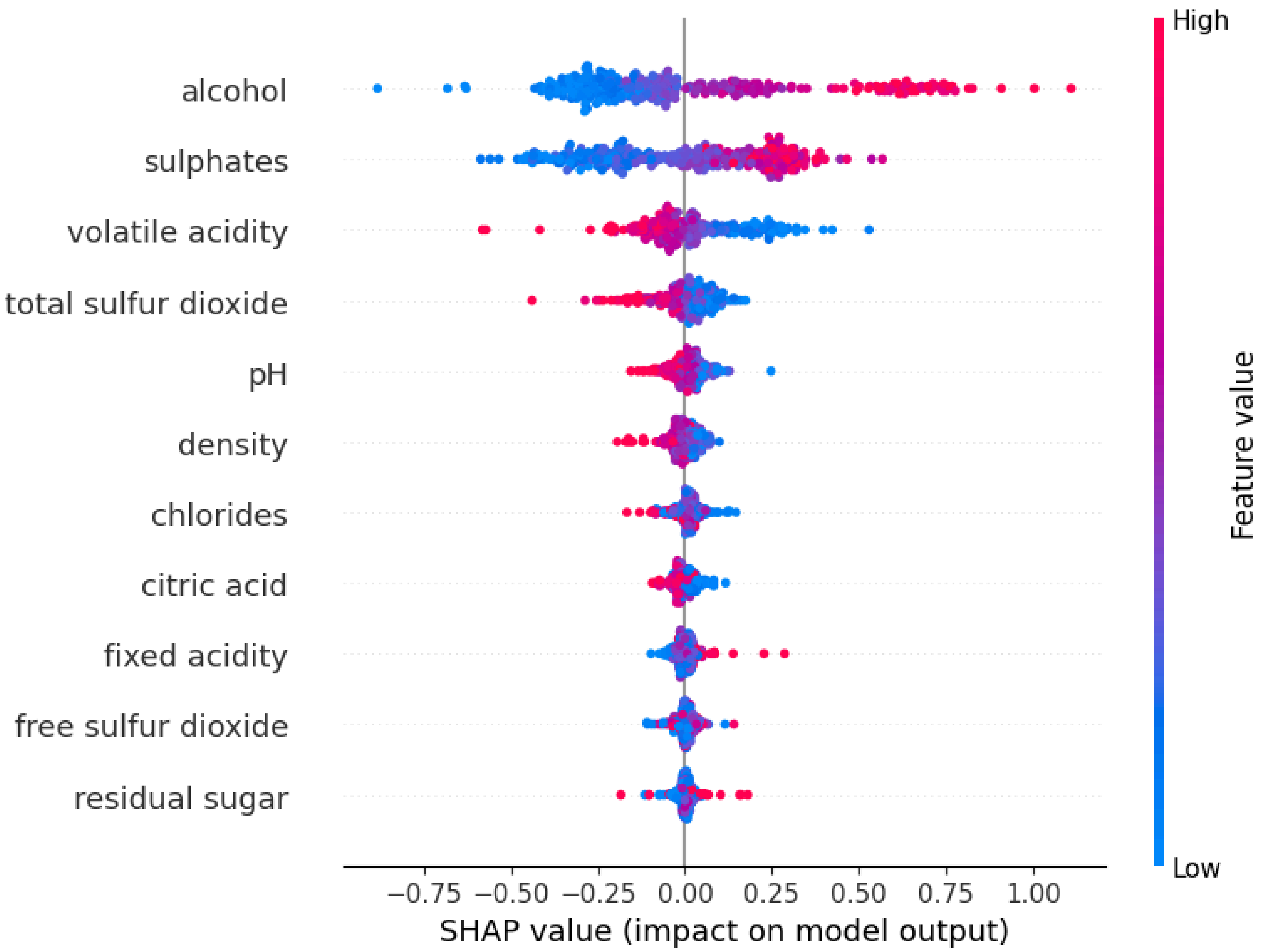
[8]:



[8]:

```
#Global Explanation
shap.summary_plot(shap_values, X_test)
```

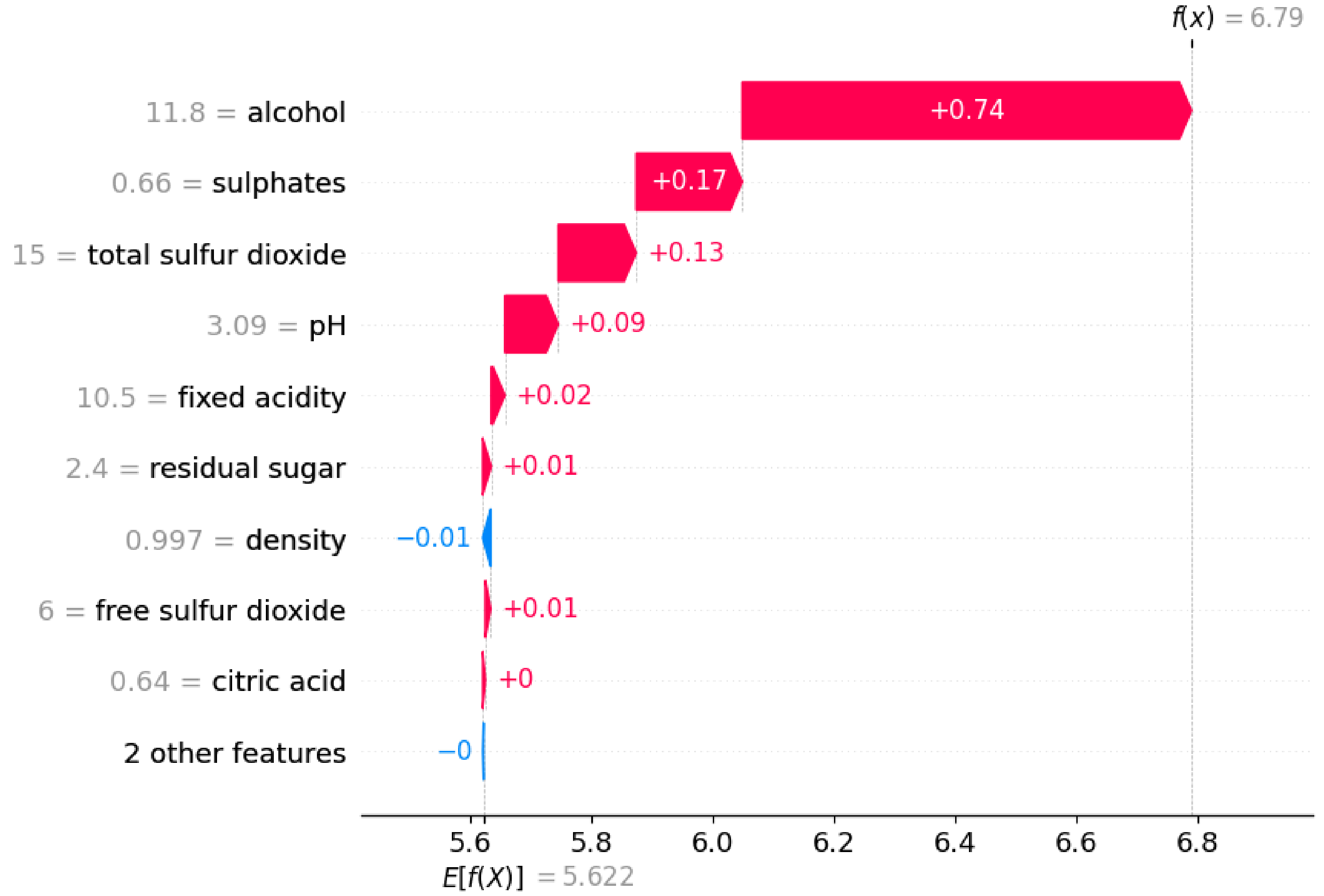
[9]:



10]:

```
#Local Explainability using waterfall plot for 50th instance
shap.plots.waterfall(shap_values[10])
```

10]:



12]:

```
#KERNEL EXPLAINER
kexp=shap.KernelExplainer(model.predict,X_train)
shap_values=kexp.shap_values(X_test.iloc[:3])
```

12]:

```
Using 1279 background data samples could cause slower run times. Consider using shap.sample(data, K) or shap.kmeans(data, K) to summarize the background as K samples.
```

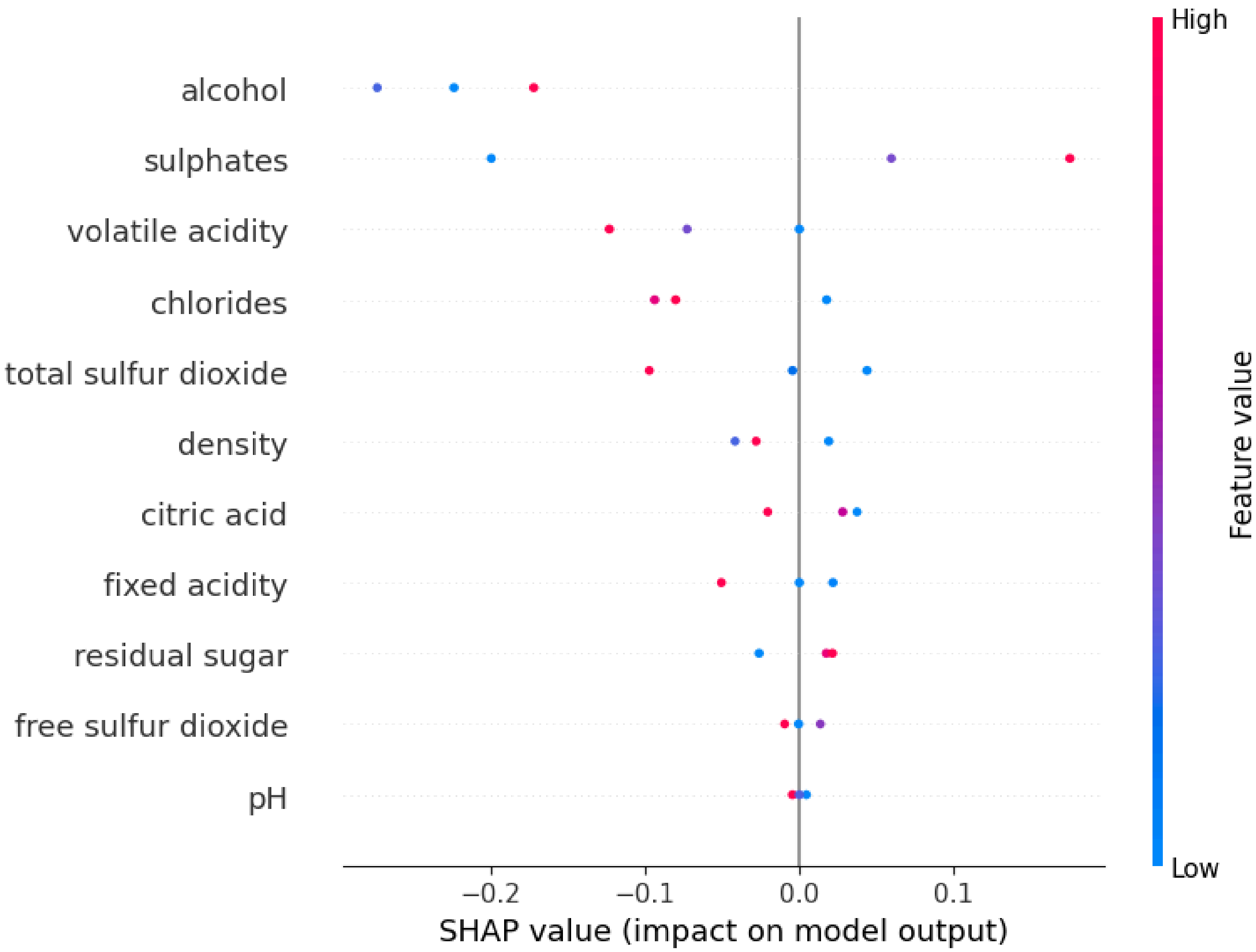
12]:

```
0%|          | 0/3 [00:00
```

18]:

```
#Global Interpretability
#X_test.iloc[:3] because we found only 3 shapely values
shap.summary_plot(shap_values,X_test.iloc[:3])
```

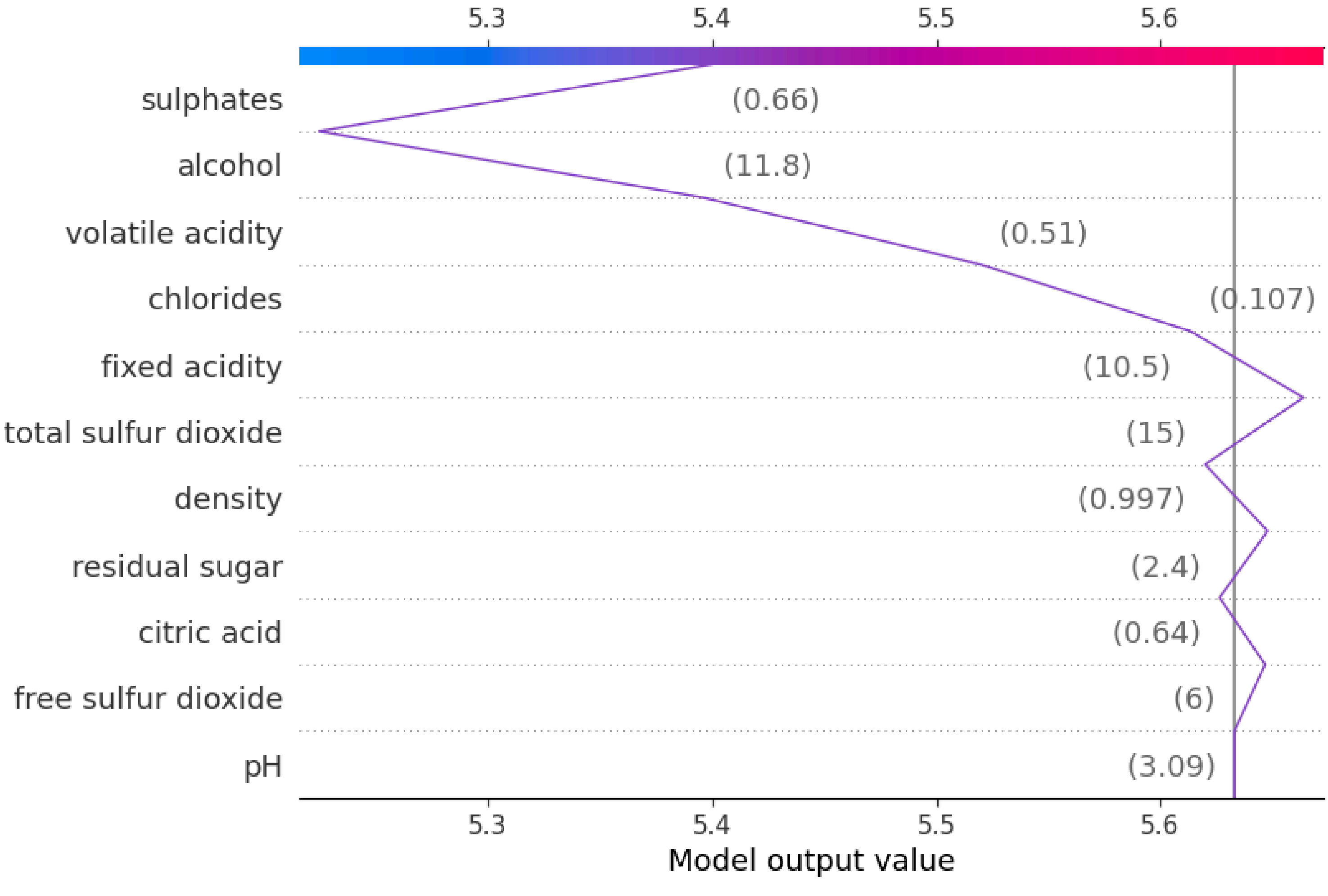
18]:



19]:

```
shap.decision_plot(kexp.expected_value, shap_values[2], X_test.iloc[10,:])
#How model predicts for 10th sample for 2nd instance if we remove it will show 3 lines i.e all 3 instances
```

19]:



SHAP ON DL MODELS

[1]:

```python
import shap
import numpy as np
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense,Flatten,MaxPool2D,Conv2D,Input
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split

# Load and preprocess dataset
full_ds = tf.keras.utils.image_dataset_from_directory(
    r'cat_dog',          # Root folder with subfolders as class names
    image_size=(224, 224),
    # color_mode='grayscale'
).map(lambda x, y: (x / 255.0, y))

# Convert dataset to numpy arrays
x_full, y_full = [], []
for images, labels in full_ds:
    x_full.append(images.numpy())
    y_full.append(labels.numpy())
x_full = np.concatenate(x_full, axis=0)
y_full = np.concatenate(y_full, axis=0)

# Train-test split
x_train, x_test, y_train, y_test = train_test_split(x_full, y_full, test_size=0.3, random_state=42)

def build_cnn(inp_shape=(224,224,3),num_cls=2):
    inp=Input(shape=inp_shape)
    x=Conv2D(16,(3,3),activation='relu')(inp)
    x=MaxPool2D((2,2))(x)
    x=Conv2D(32,(3,3),activation='relu')(x)
    x=MaxPool2D((2,2))(x)
    x=Conv2D(32,(3,3),activation='relu')(x)
    x=MaxPool2D((2,2))(x)
    x=Flatten()(x)
    x=Dense(64,activation='relu')(x)
    outputs=Dense(num_cls,activation='softmax')(x)
    model=Model(inputs=inp,outputs=outputs)
    return model

model = build_cnn()
model.summary()
# Compile and train the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
```

[1]:

```
Found 132 files belonging to 2 classes.
```

[1]:

Model: "functional"

[1]:

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 222, 222, 16)	448
max_pooling2d (MaxPooling2D)	(None, 111, 111, 16)	0
conv2d_1 (Conv2D)	(None, 109, 109, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0
conv2d_2 (Conv2D)	(None, 52, 52, 32)	9,248
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 32)	0
flatten (Flatten)	(None, 21632)	0
dense (Dense)	(None, 64)	1,384,512
dense_1 (Dense)	(None, 2)	130

[1]:

Total params: 1,398,978 (5.34 MB)

[1]:

Trainable params: 1,398,978 (5.34 MB)

[1]:

Non-trainable params: 0 (0.00 B)

[1]:

```
Epoch 1/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 8s 1s/step - accuracy: 0.4907 - loss: 0.8789
Epoch 2/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.3986 - loss: 0.8807
Epoch 3/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 0.5482 - loss: 0.6682
Epoch 4/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 0.6075 - loss: 0.6305
```

Epoch 5/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.6299 - loss: 0.6084

[1]:
[2]:

```python
img_path = 'lion.jpg'  # change this path
img = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img)
img_preprocessed = np.expand_dims(img_array, axis=0)

# Show top predicted class
background = np.ones((1, 224, 224, 3)) * 255
preds = model.predict(img_preprocessed)
explainer = shap.DeepExplainer(model, background)
shap_values = explainer.shap_values(img_preprocessed)

# 3. Visualize SHAP values for the image
plt.figure(figsize=(8, 4))
shap.image_plot(shap_values, img_preprocessed)
```

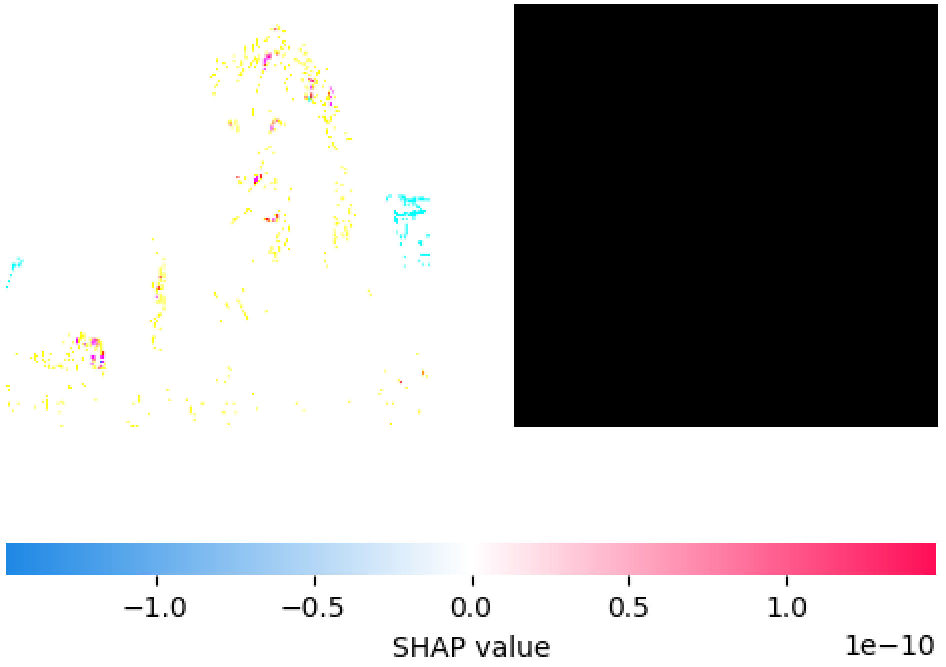
[2]:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 327ms/step
```

[2]:

```
E:\Xai_Req_Setup\Python3109\lib\site-packages\shap\explainers\_deep\deep_tf.py:94: UserWarning: Your TensorFlow version is newer than 2.4.0 and so graph support has been removed in eager mode and some static graphs may not be supported. See PR #1483 for discussion.
  warnings.warn(
E:\Xai_Req_Setup\Python3109\lib\site-packages\keras\src\models\functional.py:238: UserWarning: The structure of `inputs` doesn't match the expected structure.
Expected: keras_tensor
Received: inputs=['Tensor(shape=(1, 224, 224, 3))']
  warnings.warn(msg)
E:\Xai_Req_Setup\Python3109\lib\site-packages\keras\src\models\functional.py:238: UserWarning: The structure of `inputs` doesn't match the expected structure.
Expected: keras_tensor
Received: inputs=['Tensor(shape=(2, 224, 224, 3))']
  warnings.warn(msg)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [0.0..255.0].
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [-3.765308065339923e-10..2.8535396268125623e-10].
```

[2]:
[2]:



[3]:

```python
import cv2

# Convert SHAP values to single channel heatmap
heatmap = np.abs(shap_values[0][0]).mean(axis=-1)  # shape: (224, 224)
heatmap = cv2.resize(heatmap, (224, 224))
heatmap = (heatmap - heatmap.min()) / (heatmap.max() - heatmap.min())

# Convert original image to uint8
original_uint8 = (img_preprocessed[0] * 255).astype(np.uint8)

# Overlay heatmap on original image
plt.imshow(original_uint8)
plt.imshow(heatmap, cmap='jet', alpha=0.5)
plt.axis('off')
plt.title("SHAP Overlay")
plt.show()
```

[3]:

SHAP Overlay