Broad Functionality: **Model Bias**

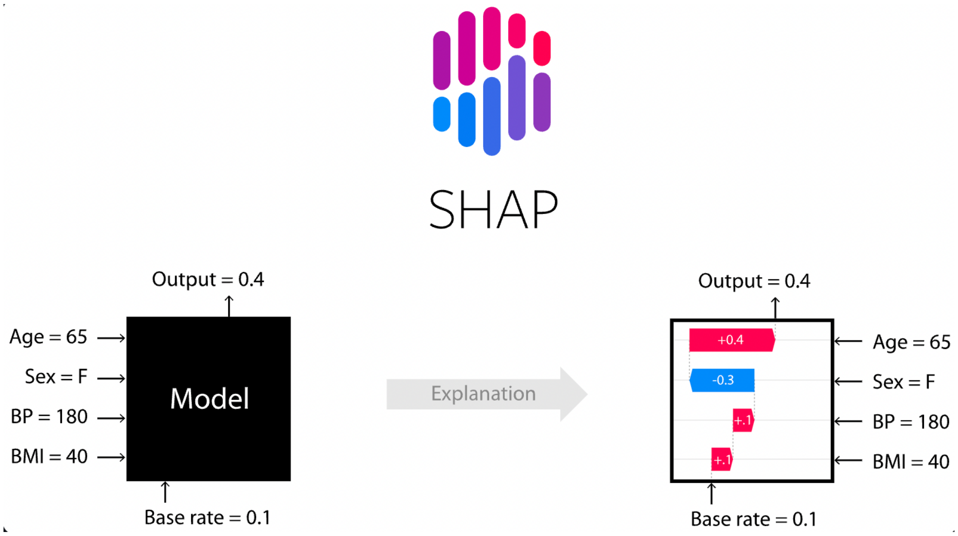
Principle: **Model Monitoring and Bias**

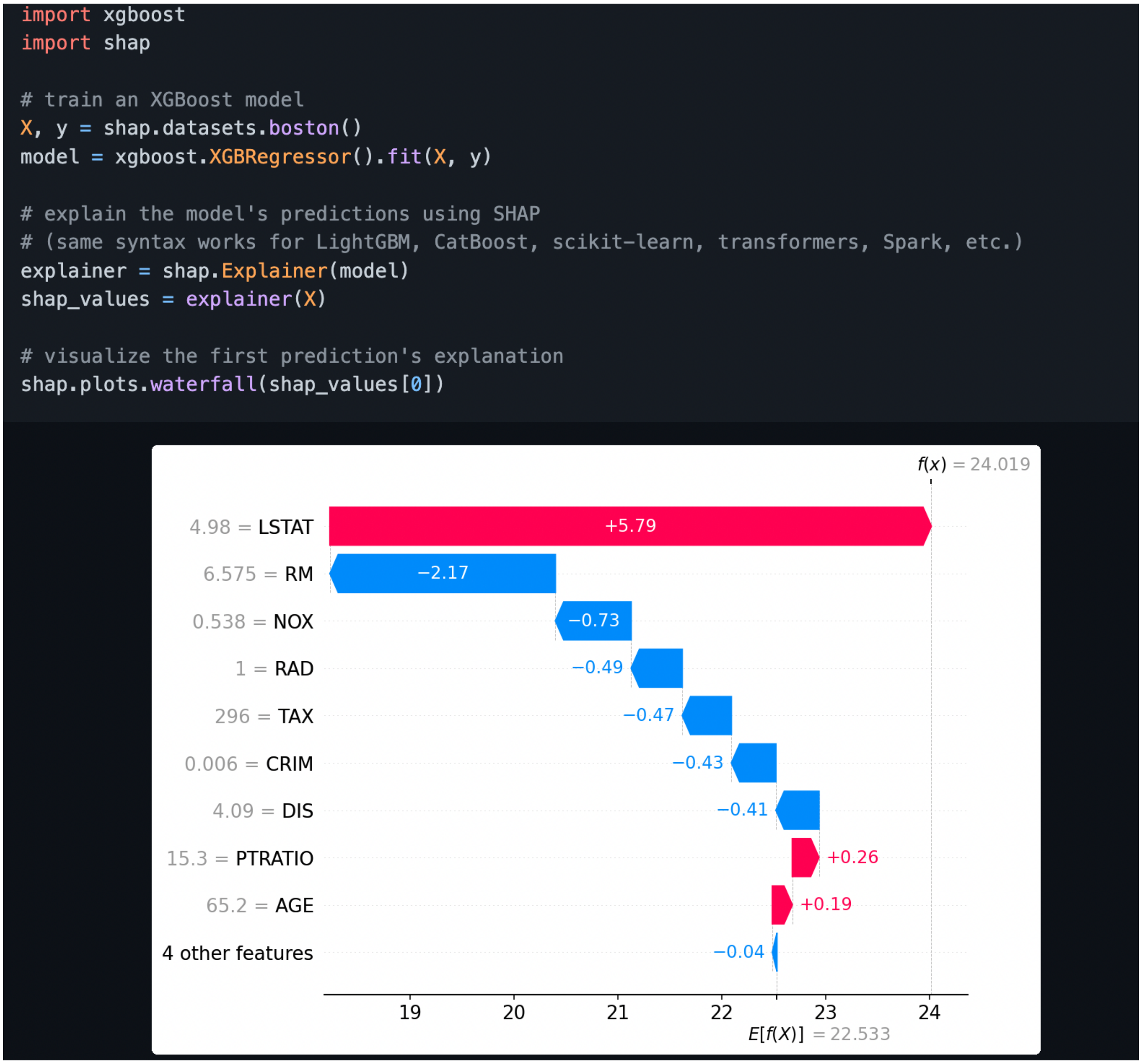
Understanding Functionality: Based on inherent bias present in the dataset used for training the model, predictions might show bias in an unacceptable way. Bias based on features such as gender, race, or region will call for actions to mitigate the potential issues that the bias in the model might create. Hence, any model in production should be monitored over such sensitive features continuously and should be capped with a threshold value on the bias. Ideally, the platform should also provide alerts and notifications when a particular variable type within a feature overshoots the threshold value.

Does MLflow supports the functionality *directly*: **No**

Does MLflow supports the functionality *indirectly*: Though the MLflow doesn't support the functionality to understand the bias in the model and to cap importance with a threshold value on the sensitive features, it does let users integrate the libraries supporting such monitoring. Popular library SHAP, i.e., Shapley Additive Explanations, can be used to understand the relative importance of the features in the model and the impact of each of these features in overall prediction over the data. Given any model, this library computes - SHAP values, from the model and these values are readily interpretable, as each value is a feature's effect on the prediction, in its units. However, this cannot be monitored dynamically at the production level or no real-time assessment of the model can be done.

Screenshots for MLflow supports the functionality *directly* or *indirectly*:





```python
import xgboost
import shap

# train an XGBoost model
X, y = shap.datasets.boston()
model = xgboost.XGBRegressor().fit(X, y)

# explain the model's predictions using SHAP
# (same syntax works for LightGBM, CatBoost, scikit-learn, transformers, Spark, etc.)
explainer = shap.Explainer(model)
shap_values = explainer(X)

# visualize the first prediction's explanation
shap.plots.waterfall(shap_values[0])
```