In [21]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import lime
from lime import lime_tabular
import shap
```

Explainable AI (XAI) is a field of study focused on making machine learning models more transparent and understandable. It aims to demystify the "black box" nature of complex models, especially deep neural networks, by providing insights into how they arrive at their decisions.
XAI is all about making AI systems easier to understand. Imagine you have a smart assistant that helps you make decisions, but you want to know why it suggests certain things. XAI provides clear explanations for the AI’s choices, so you can see the reasoning behind them. This transparency ensures the AI is working fairly and correctly, which is especially important in areas like healthcare and finance.

1. LIME, which stands for Local Interpretable Model-Agnostic Explanations, is a technique used to explain the predictions of machine learning models. It works by approximating the model locally with a simpler, interpretable model:

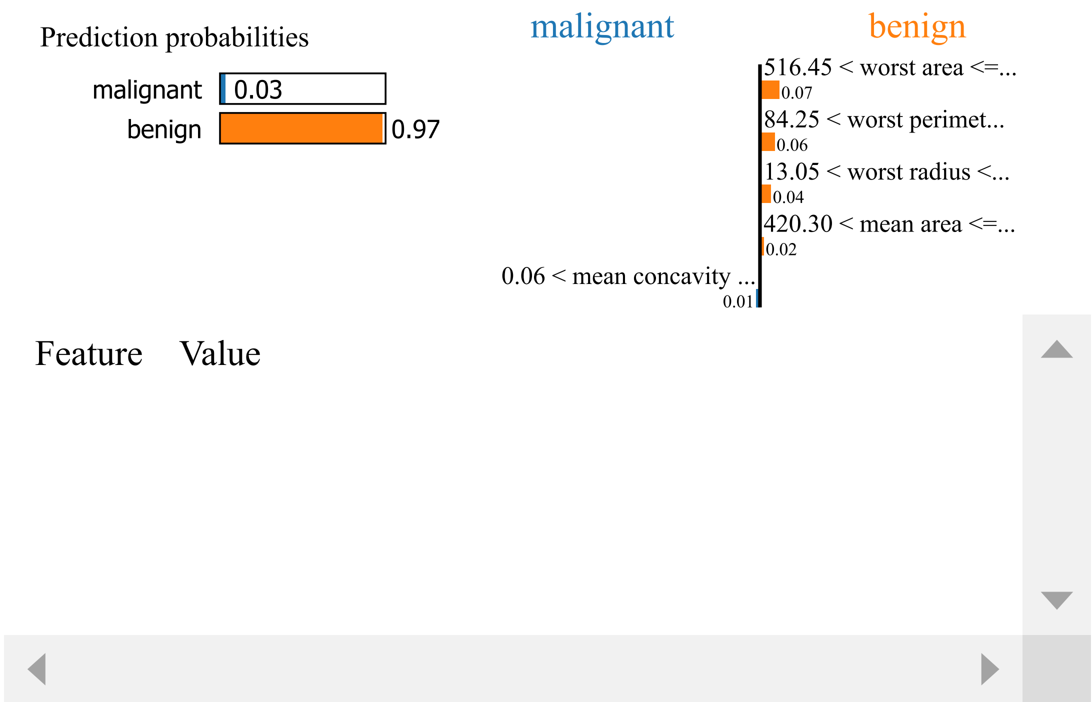
Local Explanations: LIME focuses on explaining individual predictions rather than the entire model. It perturbs the input data around the instance being explained and observes how the predictions change. Model-Agnostic: LIME can be applied to any machine learning model, regardless of its complexity or type. This makes it a versatile tool for interpreting black-box models. Interpretable Models: By fitting a simple, interpretable model (like a linear model) to the perturbed data, LIME provides insights into which features are most influential for a specific prediction.

In [45]:

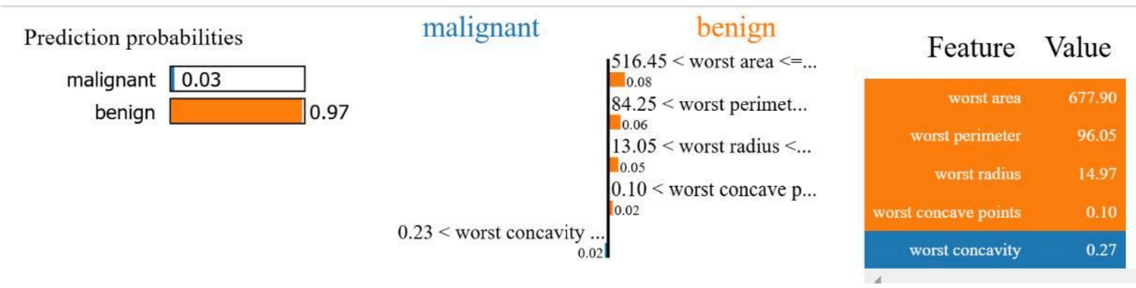
```
# Load dataset (UCI ML Breast Cancer Wisconsin (Diagnostic))
data = load_breast_cancer()
X = data.data
y = data.target
feature_names = data.feature_names
class_names = data.target_names
# class_names are ['malignant', 'benign'], representing the two possible outcome

# Display the first few rows of the dataframe
df = pd.DataFrame(data.data, columns=data.feature_names)
print(df.head(3))
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38           122.8     1001.0          0.11840
1        20.57         17.77           132.9     1326.0          0.08474
2        19.69         21.25           130.0     1203.0          0.10960

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069

   mean fractal dimension  ...  worst radius  worst texture  worst perimeter
\
0                 0.07871  ...         25.38          17.33            184.6
1                 0.05667  ...         24.99          23.41            158.8
2                 0.05999  ...         23.57          25.53            152.5

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758

[3 rows x 30 columns]
```

In [46]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Initialize LIME explainer: Create a LIME explainer for tabular data, specifyi
explainer = lime_tabular.LimeTabularExplainer(training_data=X_train,
                                              feature_names=feature_names,
                                              class_names=class_names,
                                              mode='classification')

# Choose a sample to explain
i = 0  # Index of the sample in the test set
exp = explainer.explain_instance(X_test[i], model.predict_proba, num_features=5

# Display the explanation
exp.show_in_notebook(show_table=True, show_all=False)
```



An image from explaining a different row:



We're looking at a classification model that predicts whether a tumor is malignant or benign based on certain features like worst area, perimeter, radius, etc.
The model is highly confident in predicting the tumor as benign (0.97 probability).

Each feature's impact on the prediction is shown with horizontal bars.
Longer bars indicate a stronger influence on the prediction.

The color of the bar indicates whether the feature increases (orange) or decreases (blue) the prediction. The size of the bar indicates the magnitude of the feature's impact.

Feature Contributions. For this specific instance, the model's prediction of "benign" is primarily influenced by:
Worst Area: A smaller worst area contributes to the benign prediction.
Worst Perimeter: A smaller worst perimeter also contributes to the benign prediction.
Worst Radius: Similarly, a smaller worst radius supports the benign prediction.

Do simply higher feature values mean more impact on the decision? Not necessarily. It is influenced by the model's learned relationship between the feature and the target variable (benign, malign).
If features are scaled differently (e.g., some features are normalized, others are standardized), their raw values might not be directly comparable.
Worst Area: In the context of tumor size, larger values might generally indicate a higher likelihood of malignancy, but this isn’t always linear. There could be thresholds or ranges where the impact changes.
Worst Concavity: Small values might be highly significant if they indicate a specific characteristic of benign tumors, especially when combined with other features. The model might have learned that even small changes in this feature can lead to significant changes in the predicted outcome

In [8]:

```
# Load the dataset https://www.kaggle.com/datasets/blastchar/telco-customer-chu
data = pd.read_csv("Telco-Customer-Churn.csv")
data.head(3)
```

Out[8]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	I
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	

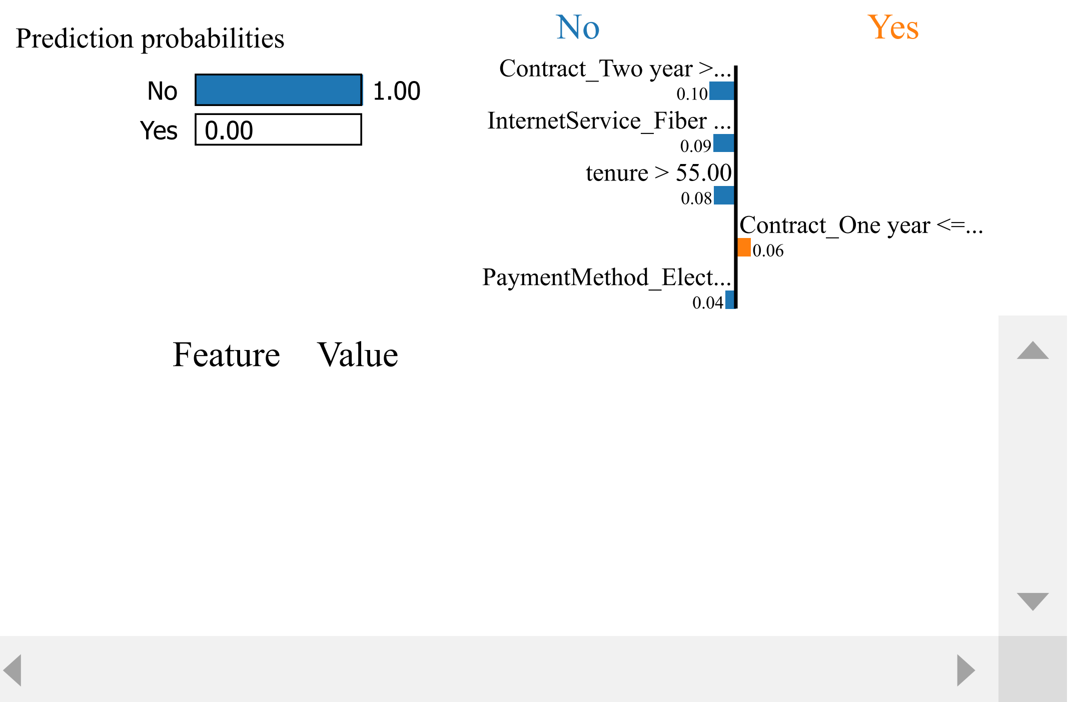
3 rows × 21 columns

In [42]:

```
# Preprocess the data
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')
data = data.dropna()

# Convert categorical variables to numeric
data = pd.get_dummies(data, columns=['gender', 'Partner', 'Dependents', 'PhoneS
                                     'InternetService', 'OnlineSecurity', 'Onli
                                     'TechSupport', 'StreamingTV', 'StreamingMo
                                     'PaperlessBilling', 'PaymentMethod'], drop_

# Define features and target
X = data.drop(columns=['customerID', 'Churn'])
y = data['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)
feature_names = X.columns
class_names = ['No', 'Yes']

# Display the first few rows of the dataframe
print(X.head(3))

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Initialize LIME explainer: Create a LIME explainer for tabular data, specifyi
explainer = lime.lime_tabular.LimeTabularExplainer(training_data=X_train.values
                                                   feature_names=feature_names,
                                                   class_names=class_names,
                                                   mode='classification')

# Choose a sample to explain
i = 0  # Index of the sample in the test set
exp = explainer.explain_instance(X_test.values[i], model.predict_proba, num_fea

# Display the explanation
exp.show_in_notebook(show_table=True, show_all=False)
```

```
   SeniorCitizen  tenure  MonthlyCharges  TotalCharges  gender_Male  \
0              0       1           29.85         29.85            0
1              0      34           56.95       1889.50            1
2              0       2           53.85        108.15            1

   Partner_Yes  Dependents_Yes  PhoneService_Yes  \
0            1               0                 0
1            0               0                 1
2            0               0                 1

   MultipleLines_No phone service  MultipleLines_Yes  ...  \
0                               1                  0  ...
1                               0                  0  ...
2                               0                  0  ...

   StreamingTV_No internet service  StreamingTV_Yes  \
0                                0                0
1                                0                0
2                                0                0

   StreamingMovies_No internet service  StreamingMovies_Yes  \
0                                    0                    0
1                                    0                    0
2                                    0                    0

   Contract_One year  Contract_Two year  PaperlessBilling_Yes  \
0                  0                  0                     1
1                  1                  0                     0
2                  0                  0                     1

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                      0                               1
1                                      0                               0
2                                      0                               0

   PaymentMethod_Mailed check
0                           0
1                           1
2                           1

[3 rows x 30 columns]
```

```
X does not have valid feature names, but RandomForestClassifier was fitted wit
h feature names
```



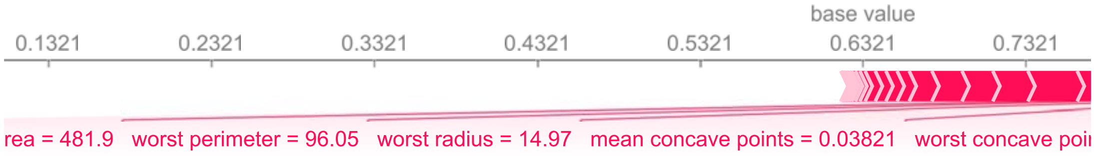
2. SHAP, which stands for SHapley Additive exPlanations, is a method used to interpret and understand the output of machine learning models. It breaks down a model's prediction to show the contribution of each feature.

In [47]:

```
# Load dataset (UCI ML Breast Cancer Wisconsin (Diagnostic))
data = load_breast_cancer()
X = data.data
y = data.target
feature_names = data.feature_names
class_names = data.target_names

df = pd.DataFrame(data.data, columns=data.feature_names)
print(df.head(3))

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Initialize SHAP explainer
explainer = shap.TreeExplainer(model)

# Choose a sample to explain
i = 0  # Index of the sample in the test set
shap_values = explainer.shap_values(X_test[i])

# Display the explanation
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], X_test[i], feature_
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38           122.8     1001.0          0.11840   
1        20.57         17.77           132.9     1326.0          0.08474   
2        19.69         21.25           130.0     1203.0          0.10960   

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419   
1           0.07864          0.0869              0.07017         0.1812   
2           0.15990          0.1974              0.12790         0.2069   

   mean fractal dimension  ...  worst radius  worst texture  worst perimeter  
\
0                 0.07871  ...         25.38          17.33            184.6   
1                 0.05667  ...         24.99          23.41            158.8   
2                 0.05999  ...         23.57          25.53            152.5   

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119   
1      1956.0            0.1238             0.1866           0.2416   
2      1709.0            0.1444             0.4245           0.4504   

   worst concave points  worst symmetry  worst fractal dimension  
0                0.2654          0.4601                  0.11890  
1                0.1860          0.2750                  0.08902  
2                0.2430          0.3613                  0.08758  

[3 rows x 30 columns]
```

Out[47]:

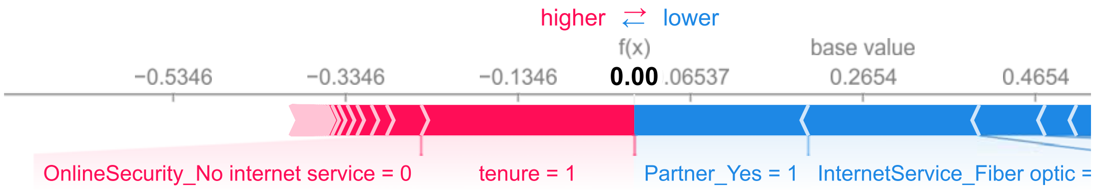


The image shows a bar graph with two bars, each made up of segments in pink and blue. These segments represent different features that affect a prediction. The pink segments indicate features that increase the prediction value, while the blue segments indicate features that decrease it. The labels below the bars show the specific features and their values, helping to understand how each feature influences the prediction.

In [48]:

```
# Load the dataset https://www.kaggle.com/datasets/blastchar/telco-customer-chur
data = pd.read_csv('Telco-Customer-Churn.csv')

# Identify non-numeric columns
non_numeric_cols = data.select_dtypes(include=['object']).columns

# Convert non-numeric columns to numeric using one-hot encoding
data_encoded = pd.get_dummies(data, columns=non_numeric_cols, drop_first=True)

# Split data into features and target variable
X = data_encoded.drop('Churn_Yes', axis=1)
y = data_encoded['Churn_Yes']

# Train a decision tree model
model = DecisionTreeClassifier()
model.fit(X, y)

# Create a SHAP explainer
explainer = shap.TreeExplainer(model)

# Explain a specific prediction
row_to_explain = X.iloc[0]  # Choose a row to explain
shap_values = explainer.shap_values(row_to_explain)

# Visualize the explanation
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], row_to_explain)
```

Out[48]:



The final prediction value in the SHAP force plot is 0. This means that after considering the contributions of all the features, the model's prediction for this particular instance is 0, which corresponds to the "No" class in a binary classification problem (no churn in a customer churn prediction model).