FEATURE IMPORTANCE IN MACHINE LEARNING

WHAT IS FEATURE IMPORTANCE?

Feature importance is about how much each input feature matters in a model. It gives each feature a score based on how important it is. If a feature has a higher score, it means it has a bigger impact on the model's predictions.

For example, Feature importance analysis in a customer churn prediction model for a subscription service might highlight factors such as subscription plan, usage frequency, and customer tenure as key indicators of churn. This insight enables targeted retention strategies, such as personalized incentives and proactive engagement, to mitigate churn and enhance customer satisfaction.

WHY FEATURE IMPORTANCE MATTERS.

Feature importance is needed for the following reasons:

- **Deciphering Models:**

  It illuminates the model’s decision-making process by pinpointing the features that heavily influence predictions.

- **Data Insights:**

  It uncovers valuable insights into the relationships and patterns embedded within the data.

- **Overfitting Mitigation:**

  Identifying and eliminating less crucial features can help curb model complexity and overfitting.

- **Boosting Generalization:**

  It can bolster the model’s capacity to generalize to unseen data by concentrating on the most crucial features.

Some real-world applications where it aids decision-making or analysis are:

1) Healthcare

   In predicting disease outcomes, feature importance identifies the most significant risk factors. For instance, in heart disease prediction, factors like cholesterol levels, age, and blood pressure might be identified as crucial.

2) Finance

In credit scoring models, feature importance can highlight the most influential factors affecting credit risk, such as income level, employment status, and credit history.

3) Marketing

In customer segmentation, feature importance can reveal the key characteristics that differentiate customer groups, such as spending habits, product preferences, or demographic information.

4) Environmental Science

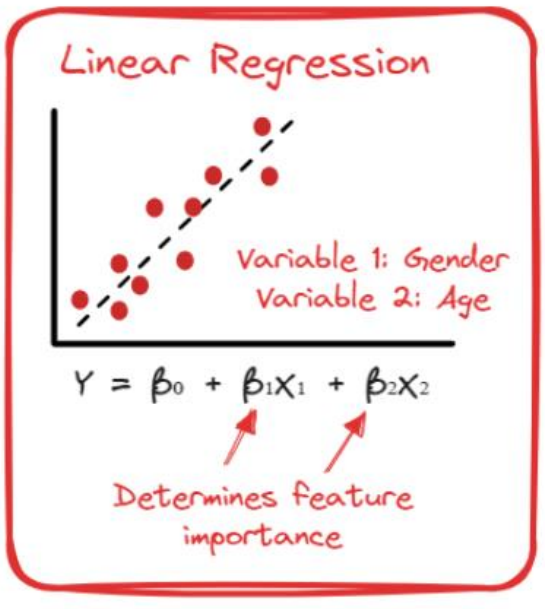
In climate change studies, feature importance can help determine the most impactful factors on global warming, such as CO2 emissions, deforestation rates, or ocean temperature changes.

METHODS FOR DETERMINING FEATURE IMPORTANCE

1) COEFFICIENTS IN LINEAR MODELS

In Linear Regression, the key output is the coefficients. These numerical values are indicative of the relationship between each feature and the outcome variable. They serve as a measure of feature importance. By considering the absolute value of these coefficients, one can rank the features. Those with the largest absolute values are deemed the most significant.

EXAMPLE:

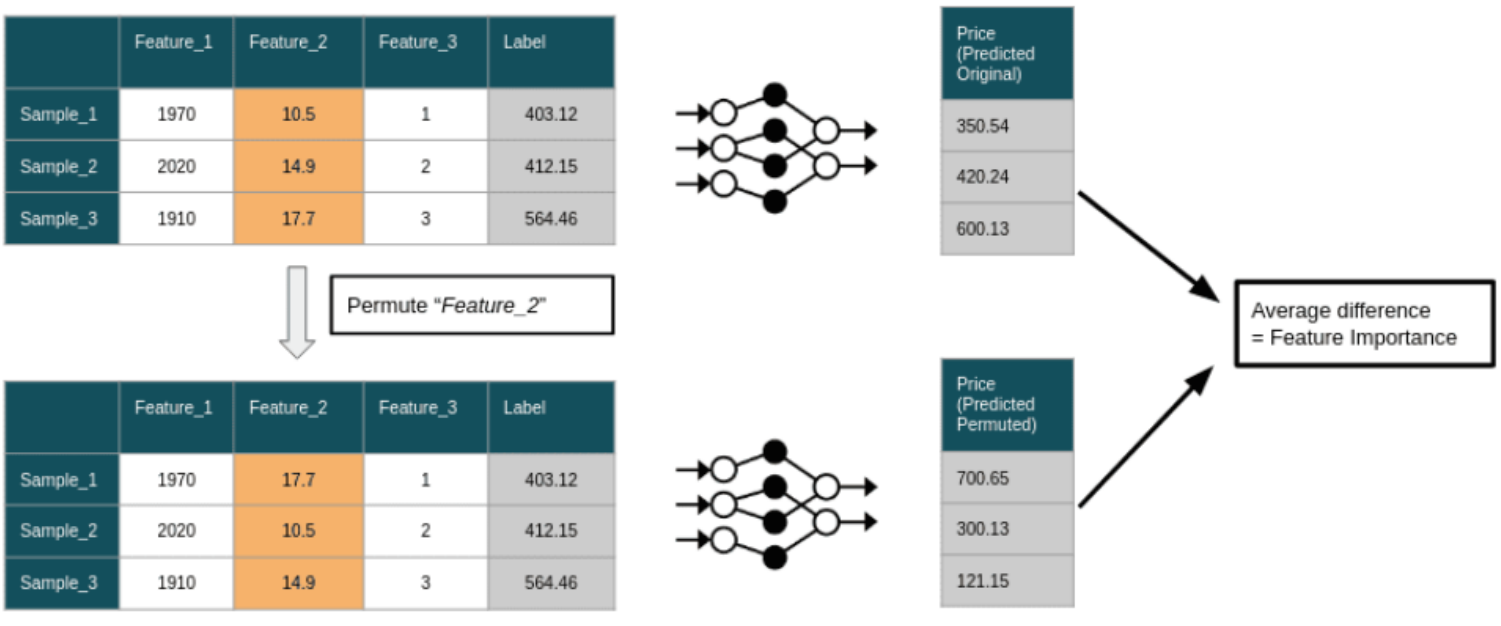


SOURCE: https://images.app.goo.gl/JFBrbfUmKxVLftex5

2) PERMUTATION IMPORTANCE

Permutation feature importance assesses a feature's impact on model effectiveness by randomly rearranging its values. If a feature is significant, randomizing its values should reduce model effectiveness. Multiple shuffling rounds may be needed for stable results.
The importance of a feature is determined by comparing model performance before and after shuffling, with the most crucial feature being the one that causes the greatest

performance drop.



SOURCE: https://images.app.goo.gl/eFtnhy7gXG7oPqzQA

3) FEATURE IMPORTANCE IN TREE-BASED MODELS

DECISION TREES

In decision tree algorithms like CART (Classification And Regression Trees), feature importance is gauged by the decrease in the criterion (Gini or entropy) used for data splits. A larger decrease indicates a more important feature, as it better separates the data into distinct groups, aiding in accurate predictions.
Scikit-learn's DecisionTreeRegressor and DecisionTreeClassifier classes can be utilized for determining feature importance via the Decision Tree algorithm

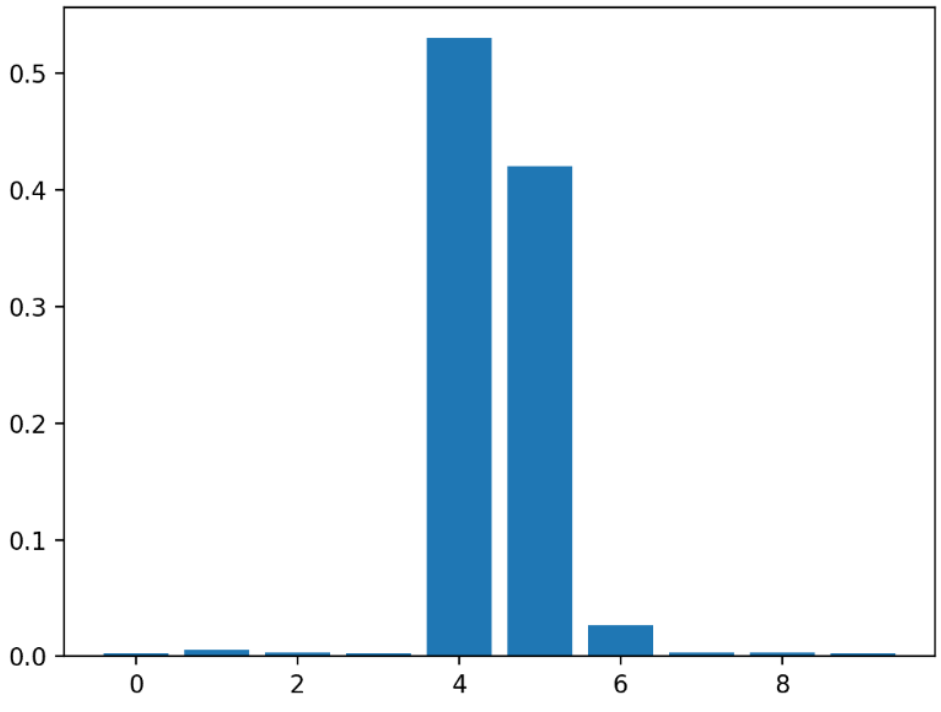
CART Regression Feature Importance

```
# importing libraries
from sklearn.datasets import make_regression
from sklearn.tree import DecisionTreeRegressor
from matplotlib import pyplot
# define dataset
X, y = make_regression(n_samples=1000, n_features=10, n_informative=5, random_state=1)
# define the model
model = DecisionTreeRegressor()
# fit the model
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
	print('Feature: %0d, Score: %.5f' % (i,v))
# plotting
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

OUTPUT:

```
Feature: 0, Score: 0.00294
Feature: 1, Score: 0.00502
Feature: 2, Score: 0.00318
Feature: 3, Score: 0.00151
Feature: 4, Score: 0.51648
Feature: 5, Score: 0.43814
Feature: 6, Score: 0.02723
Feature: 7, Score: 0.00200
Feature: 8, Score: 0.00244
Feature: 9, Score: 0.00106
```

VISUALIZATION



SOURCE: https://machinelearningmastery.com/calculate-feature-importance-with-python/

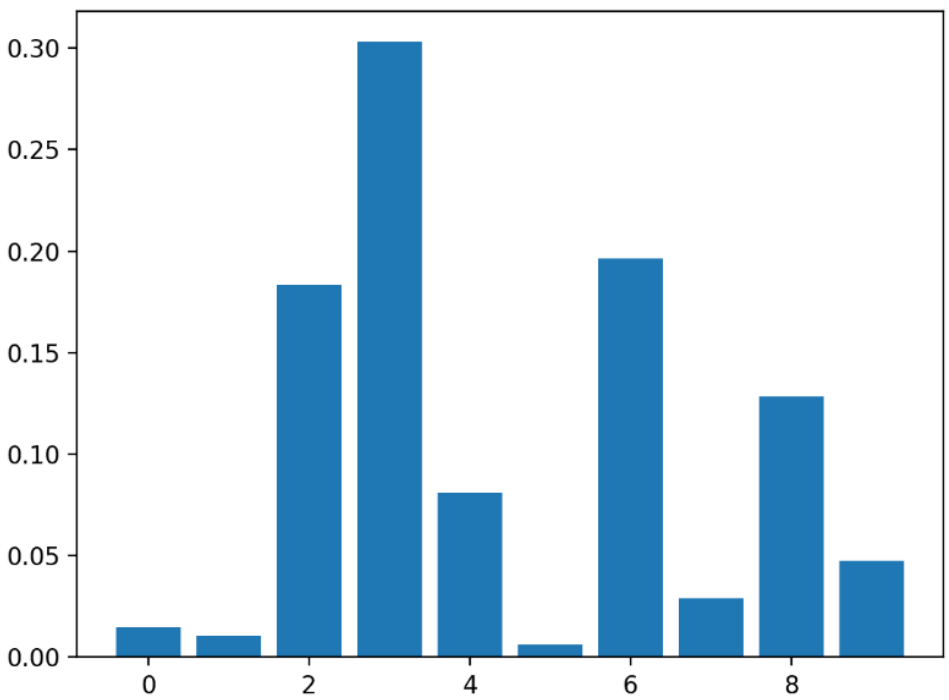
CART Classification Feature Importance

```
# importing libraries
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from matplotlib import pyplot
# define dataset
X, y = make_classification(n_samples=1000, n_features=10, n_informative=5, n_redundant=5, random_state=1)
# define the model
model = DecisionTreeClassifier()
# fit the model
model.fit(X, y)
# get importance
importance = model.feature_importances_
# summarize feature importance
for i,v in enumerate(importance):
	print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

OUTPUT:

```
Feature: 0, Score: 0.01486
Feature: 1, Score: 0.01029
Feature: 2, Score: 0.18347
Feature: 3, Score: 0.30295
Feature: 4, Score: 0.08124
Feature: 5, Score: 0.00600
Feature: 6, Score: 0.19646
Feature: 7, Score: 0.02908
Feature: 8, Score: 0.12820
Feature: 9, Score: 0.04745
```

VISUALIZATION



SOURCE: https://machinelearningmastery.com/calculate-feature-importance-with-python/

RANDOM FORESTS

Scikit-learn's RandomForestRegressor and RandomForestClassifier classes can be utilized for determining feature importance via the Random Forest algorithm. Once the model is trained, the feature_importances_ property can be accessed to obtain the relative importance scores of each feature.

CODE IMPLEMENTATION (RandomForestRegressor)

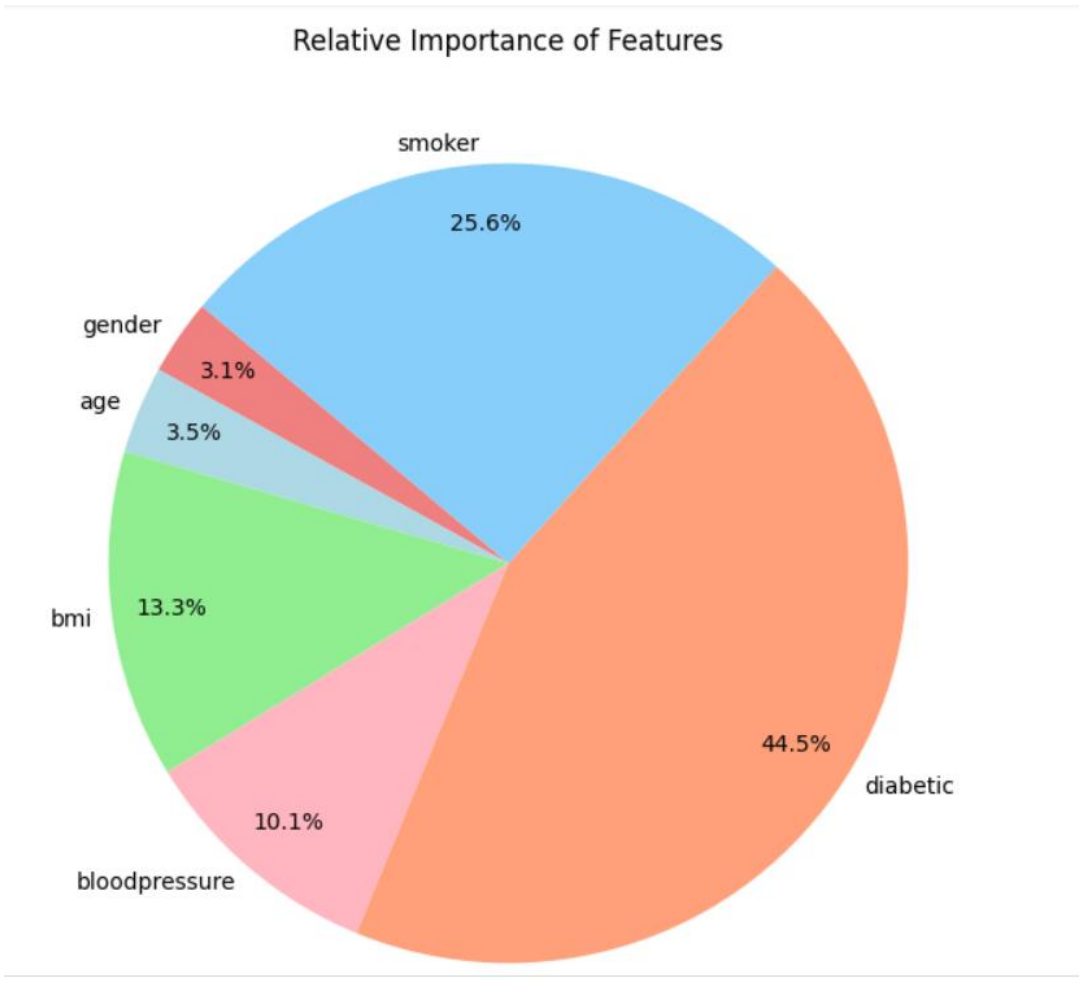
```python
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt

# Create a RandomForest model
rf_model = RandomForestRegressor()

# Fit the model to your data
rf_model.fit(X,y)  # Use your full feature set (X) and target variable (y)

# Get feature importances
feature_importances = rf_model.feature_importances_

# Get feature names (assuming X is a DataFrame with column names)
new_feature_names = ['age', 'gender', 'bmi', 'bloodpressure', 'diabetic', 'smoker']

# Sort features by importance
sorted_feature_importance = sorted(zip(new_feature_names, feature_importances), key=lambda x: x[1], reverse=True)

# Unpack feature names and importances
new_feature_names, feature_importances = zip(*sorted_feature_importance)
```

OUTPUT:

```
sorted_feature_importance
```

```
[('age', 0.5528750052677925),
 ('gender', 0.4470772962803024),
 ('bmi', 1.1715530422436737e-05),
 ('bloodpressure', 1.0993287443563078e-05),
 ('diabetic', 5.8727688942643335e-06),
 ('smoker', 4.097323927359389e-06)]
```

VISUALIZATION



SOURCE: https://colab.research.google.com/drive/1TX2P8forndm6BCvKIndT-QdfXTKFYb9g

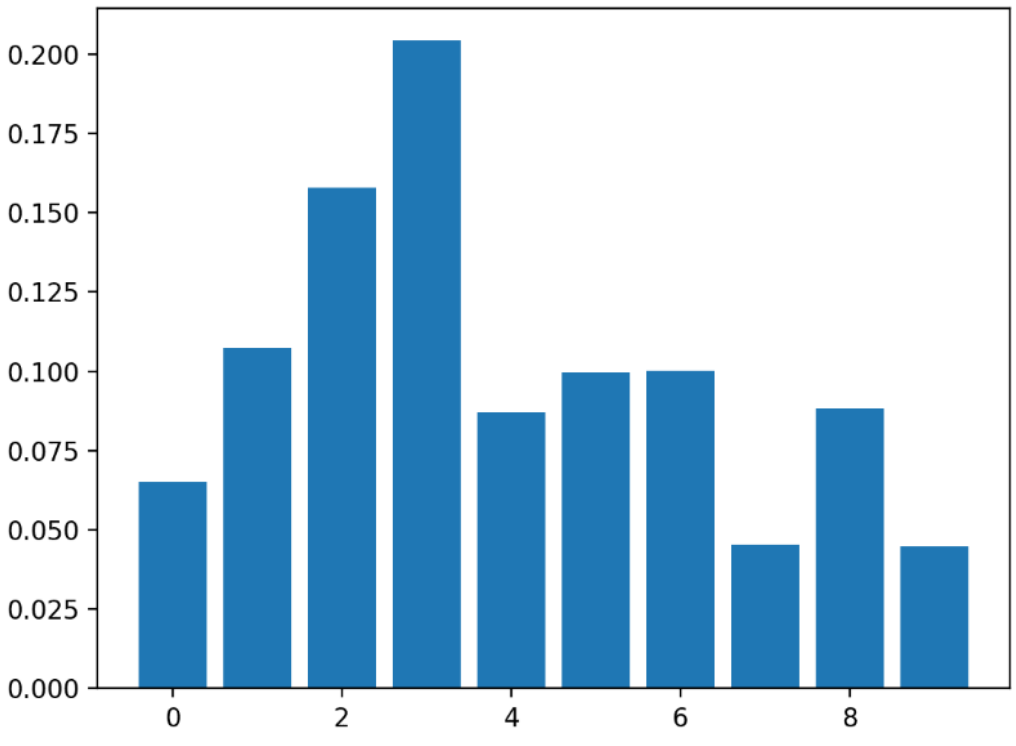
RANDOM FOREST CLASSIFIER

```
# importing libraries

from sklearn.datasets import make_classification

from sklearn.ensemble import RandomForestClassifier

from matplotlib import pyplot

# define dataset

X, y = make_classification(n_samples=1000, n_features=10, n_informative=5, n_redundant=5,

random_state=1)

# define the model

model = RandomForestClassifier()

# fit the model

model.fit(X, y)

# get importance

importance = model.feature_importances_

# summarize feature importance

for i,v in enumerate(importance):

        print('Feature: %0d, Score: %.5f' % (i,v))

# plot feature importance

pyplot.bar([x for x in range(len(importance))], importance)

pyplot.show()
```

OUTPUT:

```
Feature: 0, Score: 0.06523
Feature: 1, Score: 0.10737
Feature: 2, Score: 0.15779
Feature: 3, Score: 0.20422
Feature: 4, Score: 0.08709
Feature: 5, Score: 0.09948
Feature: 6, Score: 0.10009
Feature: 7, Score: 0.04551
Feature: 8, Score: 0.08830
Feature: 9, Score: 0.04493
```

VISUALIZATION



SOURCE: https://machinelearningmastery.com/calculate-feature-importance-with-python/

4) SHAP (SHapley Additive exPlanations) VALUES

SHAP values help us understand how each element or feature influences the outcome of a model's guess.

These values, which are rooted in game theory, give each feature a score that signifies its importance in the model. If a feature's SHAP value is positive, it boosts the model's prediction, while a negative value pulls it down. The size of the SHAP value tells us the strength of that effect.

After splitting train and test data, the following code can be used for training and prediction

EXAMPLE:

```
# Train a machine learning model

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

clf.fit(X_train, y_train)

# Make prediction on the testing data

y_pred = clf.predict(X_test)

# Classification Report

print(classification_report(y_pred, y_test))
```

OUTPUT:

	precision	recall	f1-score	support
0	0.97	0.96	0.97	815
1	0.79	0.82	0.80	130
accuracy			0.94	945
macro avg	0.88	0.89	0.88	945
weighted avg	0.94	0.94	0.94	945

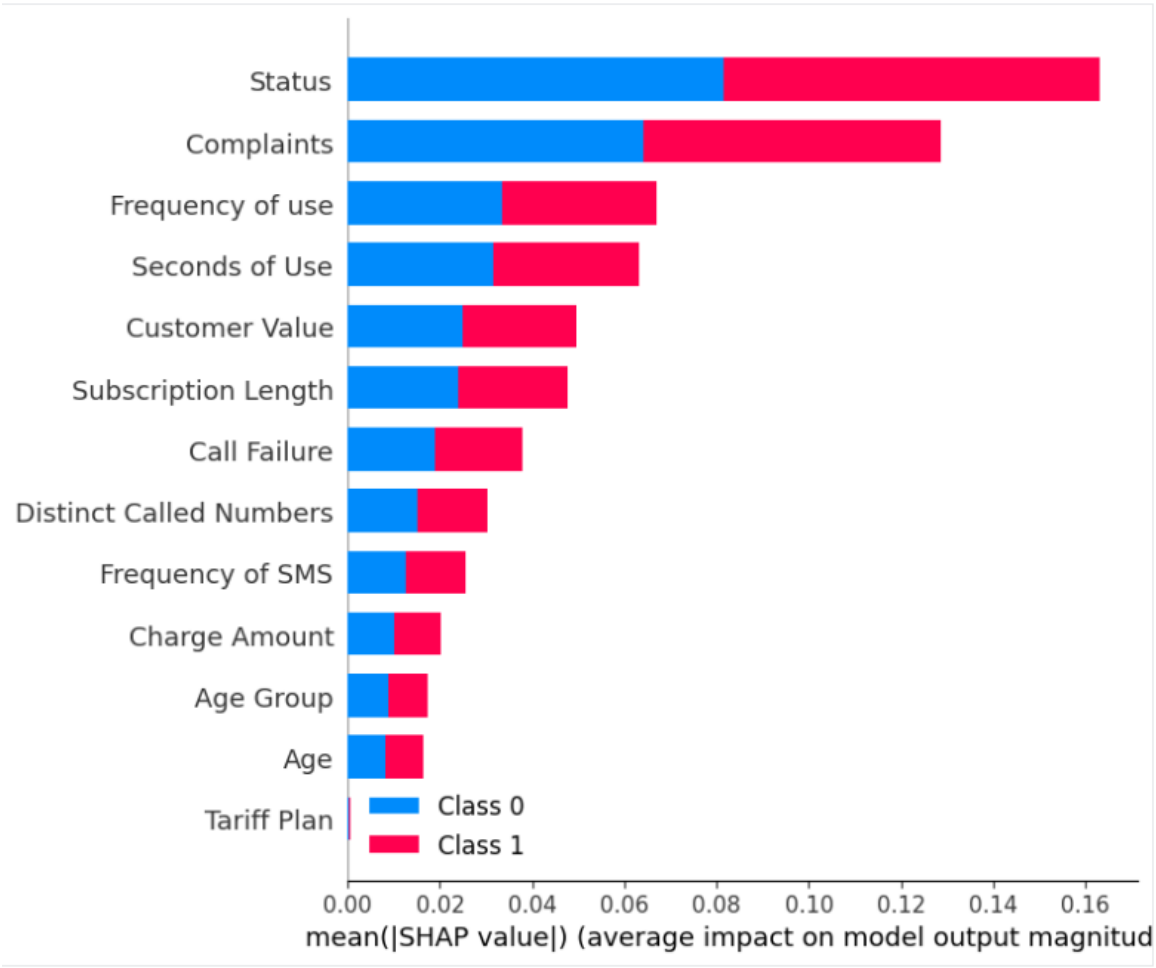
SETTING UP A SHAP EXPLAINER

We're essentially setting up an 'explainer' by feeding it a model based on random forest classification. Then, we're using this 'explainer' to compute the SHAP values with the help of a test dataset. Here's how we might express that in code:

```
explainer = shap.Explainer(clf)

shap_values = explainer.shap_values(X_test)
```

SUMMARY PLOT USING SHAP VALUES AND TEST SET

```
shap.summary_plot(shap_values, X_test)
```



SOURCE: https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability

- The (Y-axis) of the graph lists the names of the features, arranged from the most important at the top to the least important at the bottom.
- The (X-axis) shows the SHAP value, which tells us how much the log odds change.
- The color of each dot shows the value of the feature it represents - red for high values and blue for low values.
- Each dot stands for a single row of data from the original dataset.

INTERPRETABILITY OF IMPORTANCE:

Imagine running a lemonade stand with data on every cup sold, including weather, time, and customer type. You want to know what factors drive sales. Similarly, in a hospital dataset, understanding which symptoms bring patients back and how diseases affect profit is crucial. This process, known as interpretability, helps identify key predictive factors.

CONCLUSION:

Understanding the most influential parts of machine learning is essential for informed decision-making. Utilizing methods that highlight feature importance enables smart choices in feature selection, model explanation, and improvement. This understanding not only enhances model effectiveness and comprehension but also facilitates resource optimization, meaningful discoveries, and improved real-world outcomes.