Permutation importance

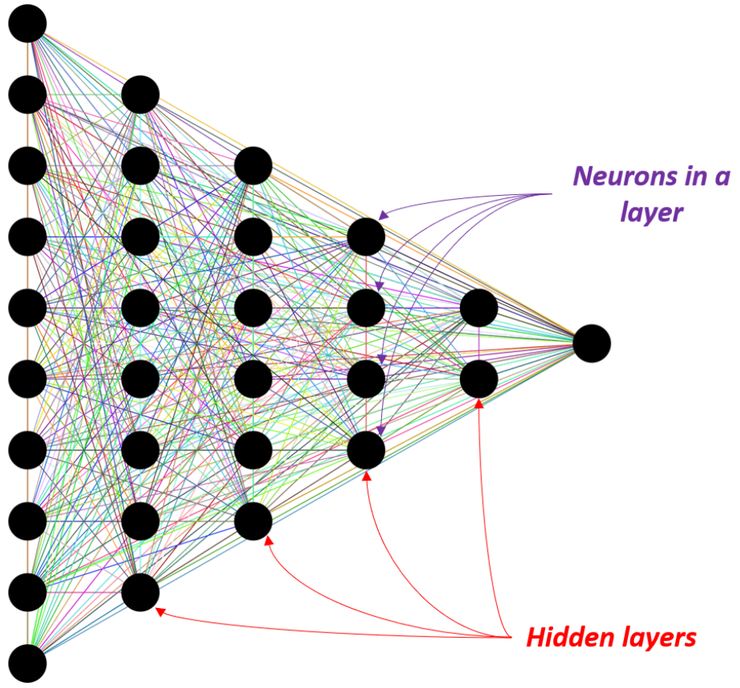
EXPLAINABLE AI IN PYTHON

Fouad Trad
Machine Learning Engineer

Shuffling notes to determine instrument's importance

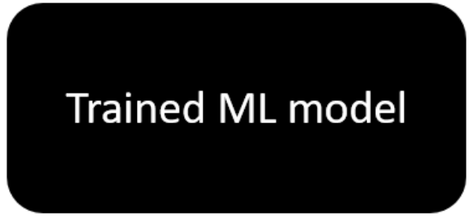
Permutation importance

- Model-agnostic method
- Assesses feature importance
  - Measures effect of feature shuffling on performance
- Highly versatile



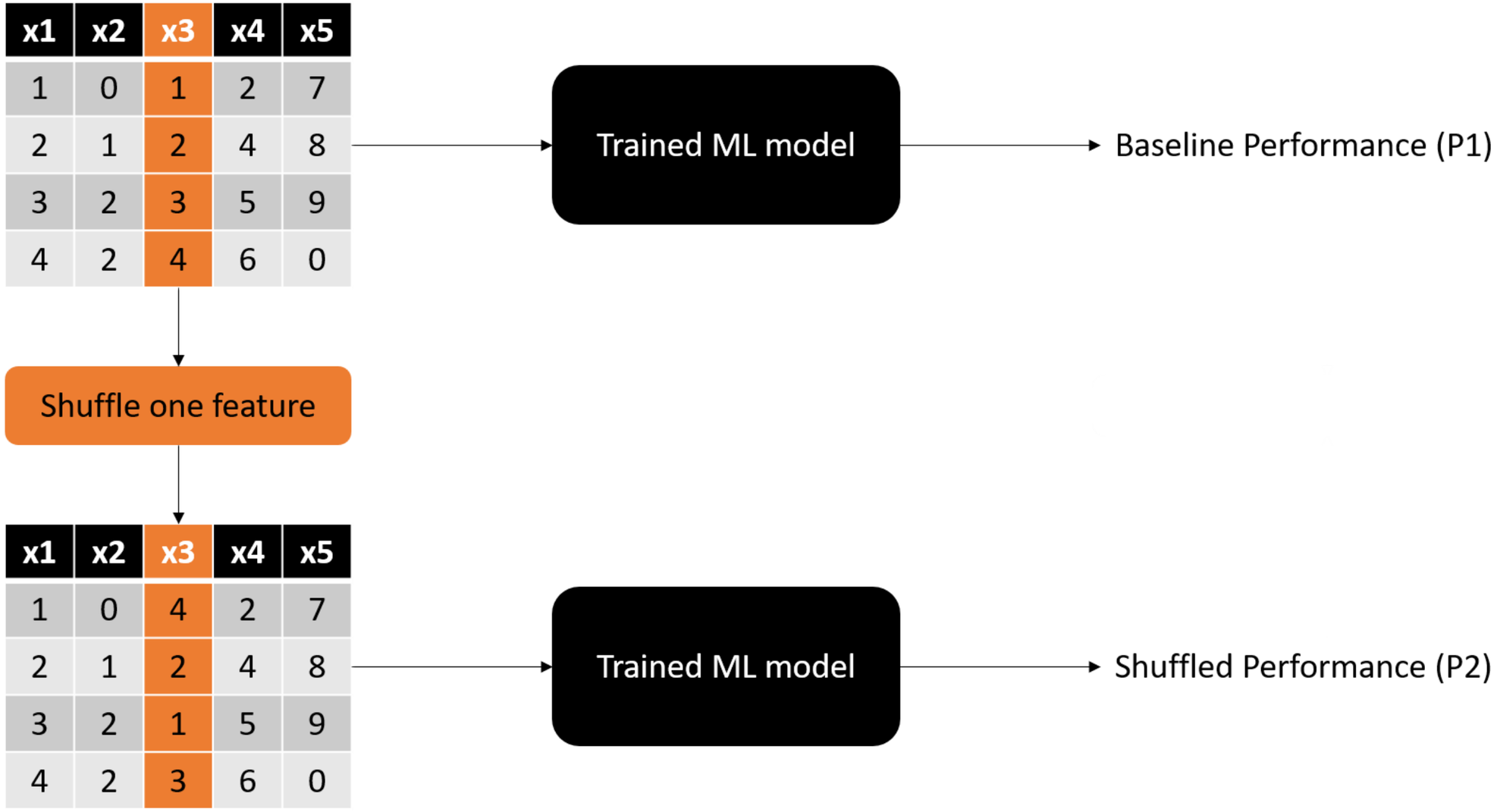
Permutation importance in action

x1	x2	x3	x4	x5

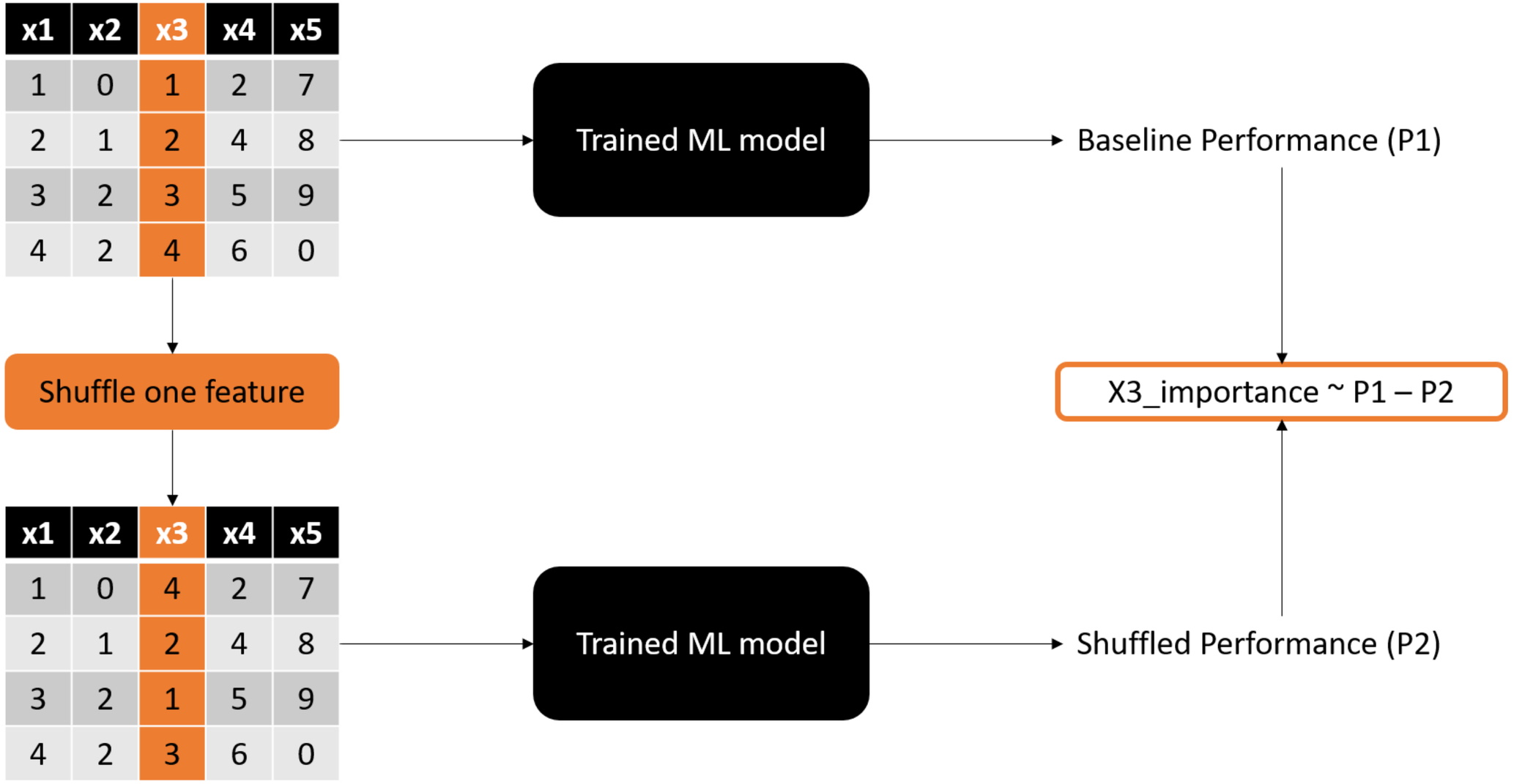


Permutation importance in action

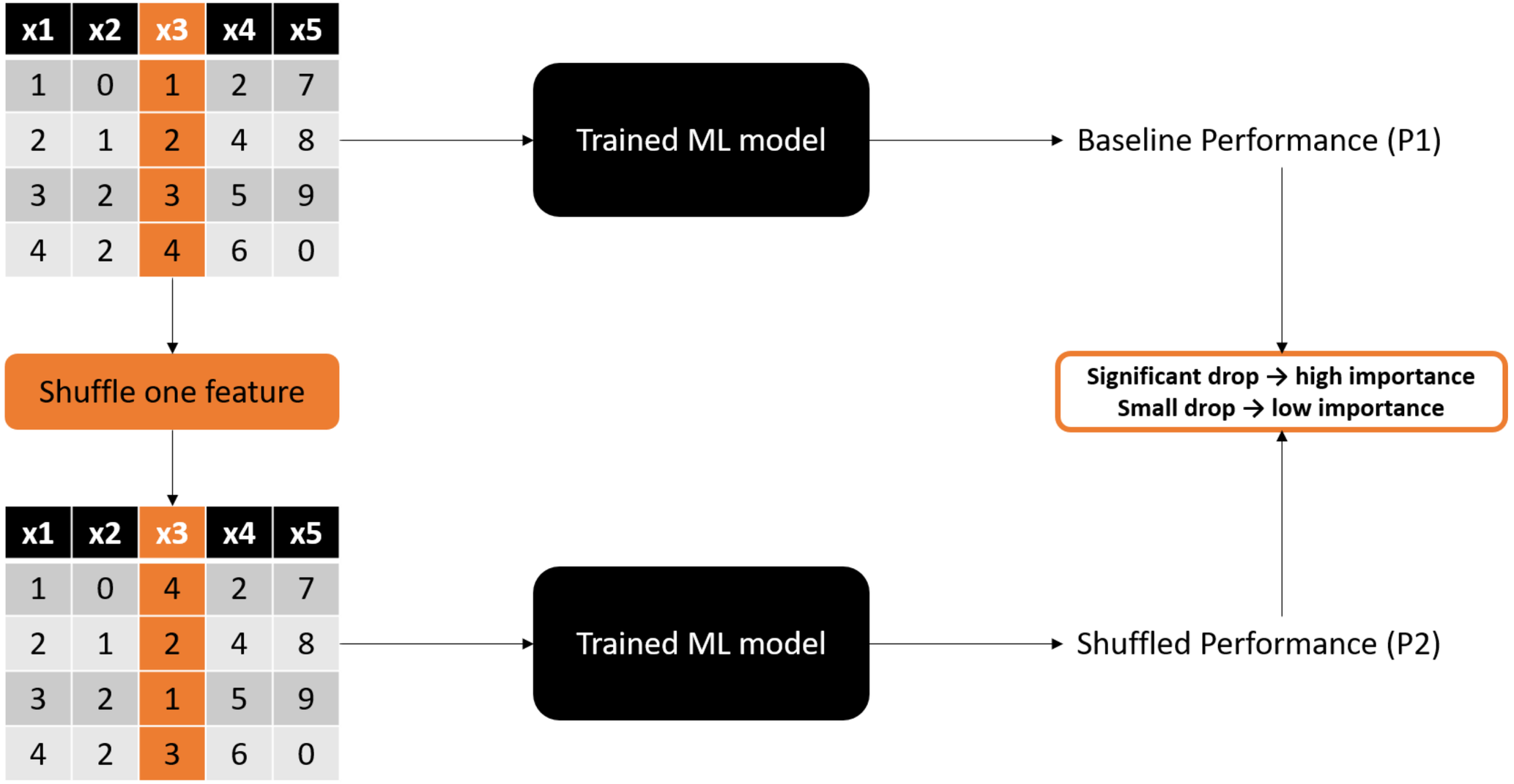
Permutation importance in action



Permutation importance in action



Permutation importance in action



Admissions dataset

GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Chance of Admit	Accept
337	118	4	4.5	4.5	9.65	0.92	1
324	107	4	4	4.5	8.87	0.76	1
316	104	3	3	3.5	8	0.72	1
322	110	3	3.5	2.5	8.67	0.8	1
314	103	2	2	3	8.21	0.45	0

The data exists in: `X_train` , `y_train`

MLPClassifier

```
from sklearn.neural_network import MLPClassifier
model = MLPClassifier(hidden_layer_sizes=(10,10))
model.fit(X_train, y_train)
```

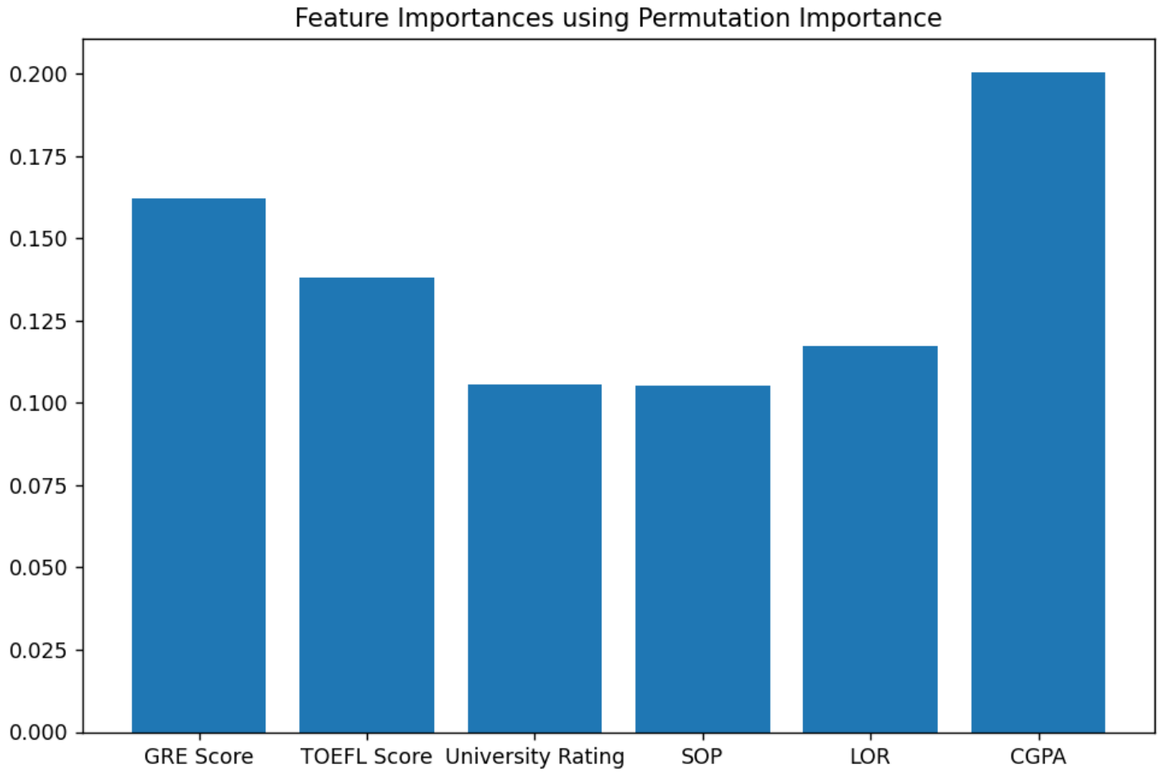
Permutation importance

```
from sklearn.inspection import permutation_importance
result = permutation_importance(model,
                                X_train, y_train,
                                n_repeats=10,
                                random_state=42,
                                scoring='accuracy')

print(result.importances_mean)
```

```
[0.16213568 0.13831658 0.10575377 0.10522613 0.11741206 0.20072864]
```

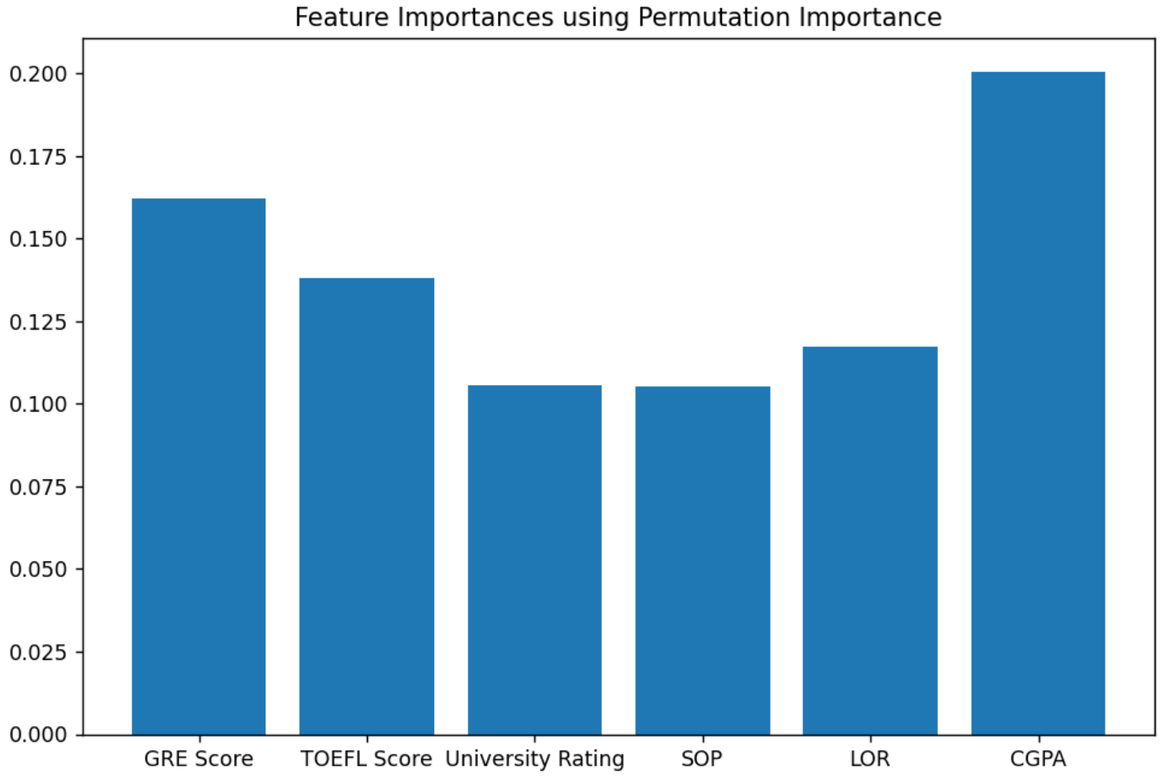
Visualizing importance

```
import matplotlib.pyplot as plt
plt.bar(X_train.columns,
        result.importances_mean)
```



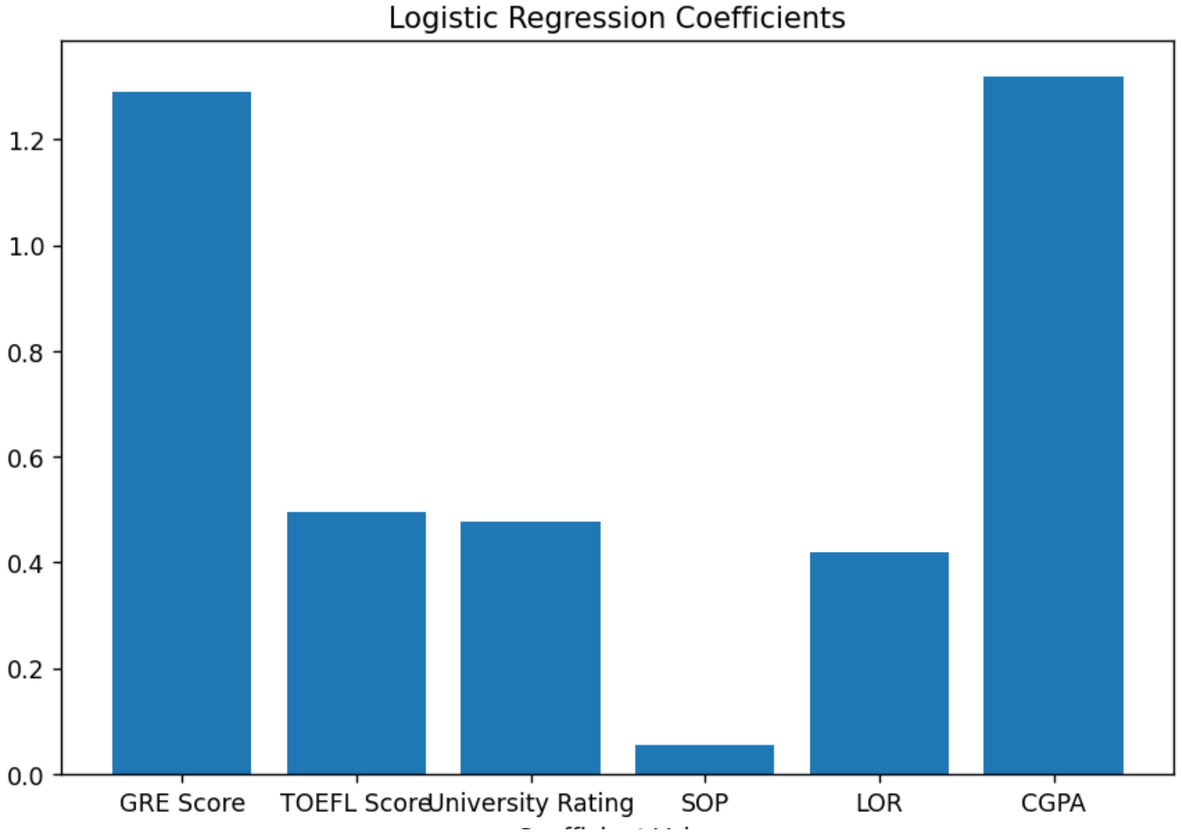
Comparison with model-specific approaches

```
import matplotlib.pyplot as plt
plt.bar(X_train.columns,
        result.importances_mean)
```



Logistic regression

```
plt.bar(X_train.columns, np.abs(log_reg.coef_[0]))
```



Let's practice!

EXPLAINABLE AI IN PYTHON

SHAP explainability

EXPLAINABLE AI IN PYTHON

Fouad Trad
Machine Learning Engineer

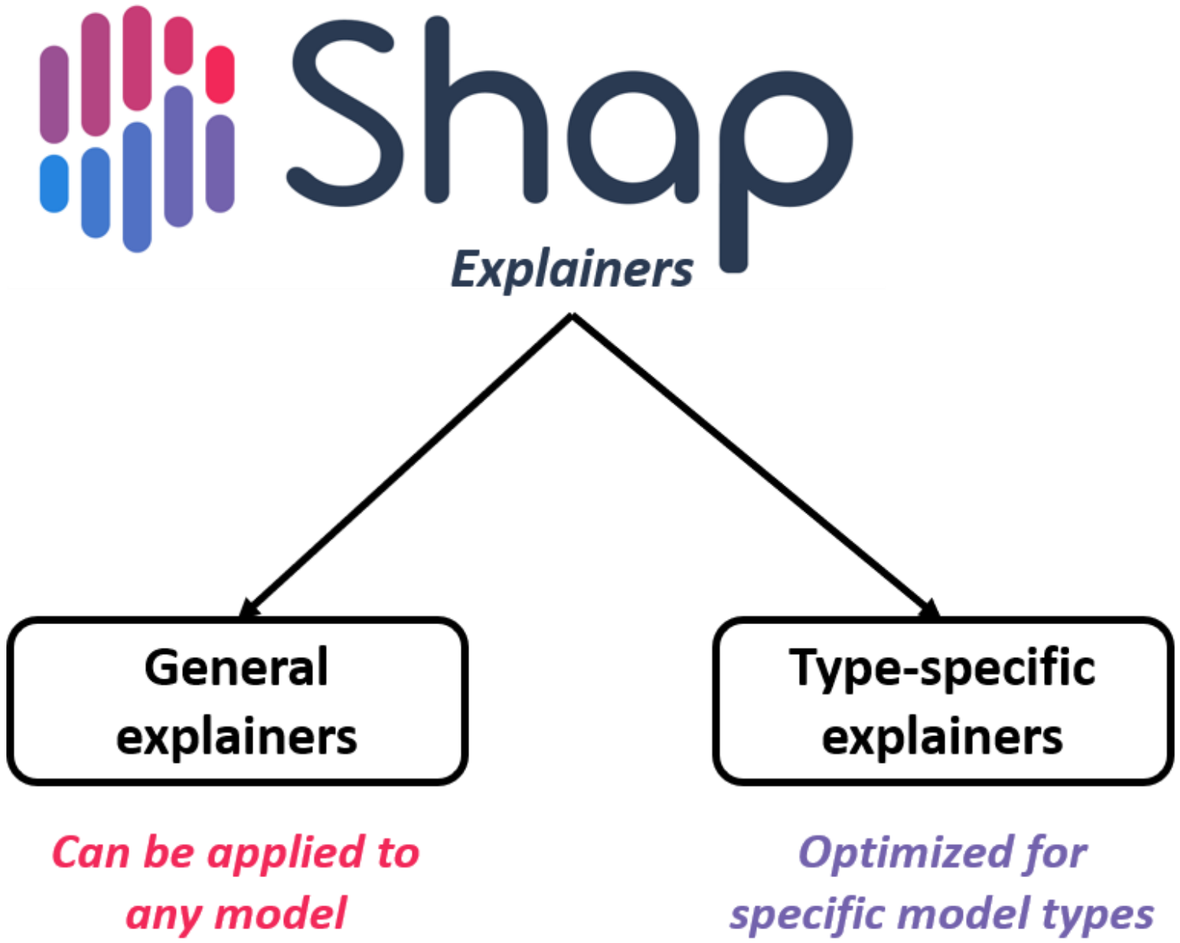
Introduction to SHAP

- **SHAP** → **SH**apley **A**dditive ex**P**lanations
- Model-agnostic technique
- Uses shapely values from game theory
- SHAP values → quantify feature contributions to predictions

Splitting profit among band members

SHAP explainers

- Compute SHAP values
- Tree explainers → tree-based models



Admissions dataset

GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Chance of Admit	Accept
337	118	4	4.5	4.5	9.65	0.92	1
324	107	4	4	4.5	8.87	0.76	1
316	104	3	3	3.5	8	0.72	1
322	110	3	3.5	2.5	8.67	0.8	1
314	103	2	2	3	8.21	0.45	0

- `rf_reg` : predicts chance of admit
- `rf_class` : predicts acceptance
- `X` : dataframe containing features

Deriving SHAP values

```
import shap
```

REGRESSION

```
explainer_reg = shap.TreeExplainer(rf_reg)
shap_values_reg = explainer_reg.shap_values(X)
```

CLASSIFICATION

```
explainer_class = shap.TreeExplainer(rf_class)
shap_values_class = explainer_class.shap_values(X)
```

Understanding SHAP output

Regression

```
print(shap_values_reg.shape)
```

```
(400, 6)
```

Features

	X1	X2	X3	X4	X5	X6
Samples	-0.05	0.00	0.00	-0.00	0.01	-0.04
	0.03	0.01	-0.01	-0.00	0.00	-0.02
	-0.03	-0.00	-0.00	0.00	0.01	-0.07
	-0.00	0.00	-0.01	-0.00	0.00	0.05
	-0.03	-0.01	-0.00	-0.01	-0.02	-0.13
	-0.01	0.00	0.00	0.00	0.01	-0.02
	-0.01	0.00	-0.00	-0.02	0.00	-0.04
	-0.05	-0.02	-0.00	-0.00	-0.01	-0.22

Understanding SHAP output

Regression

```
print(shap_values_reg.shape)
```

```
(400, 6)
```

Features

X1	X2	X3	X4	X5	X6
-0.05	0.00	0.00	-0.00	0.01	-0.04
0.03	0.01	-0.01	-0.00	0.00	-0.02
-0.03	-0.00	-0.00	0.00	0.01	-0.07
-0.00	0.00	-0.01	-0.00	0.00	0.05
-0.03	-0.01	-0.00	-0.01	-0.02	-0.13
-0.01	0.00	0.00	0.00	0.01	-0.02
-0.01	0.00	-0.00	-0.02	0.00	-0.04
-0.05	-0.02	-0.00	-0.00	-0.01	-0.22

Samples

Classification

```
print(shap_values_class.shape)
```

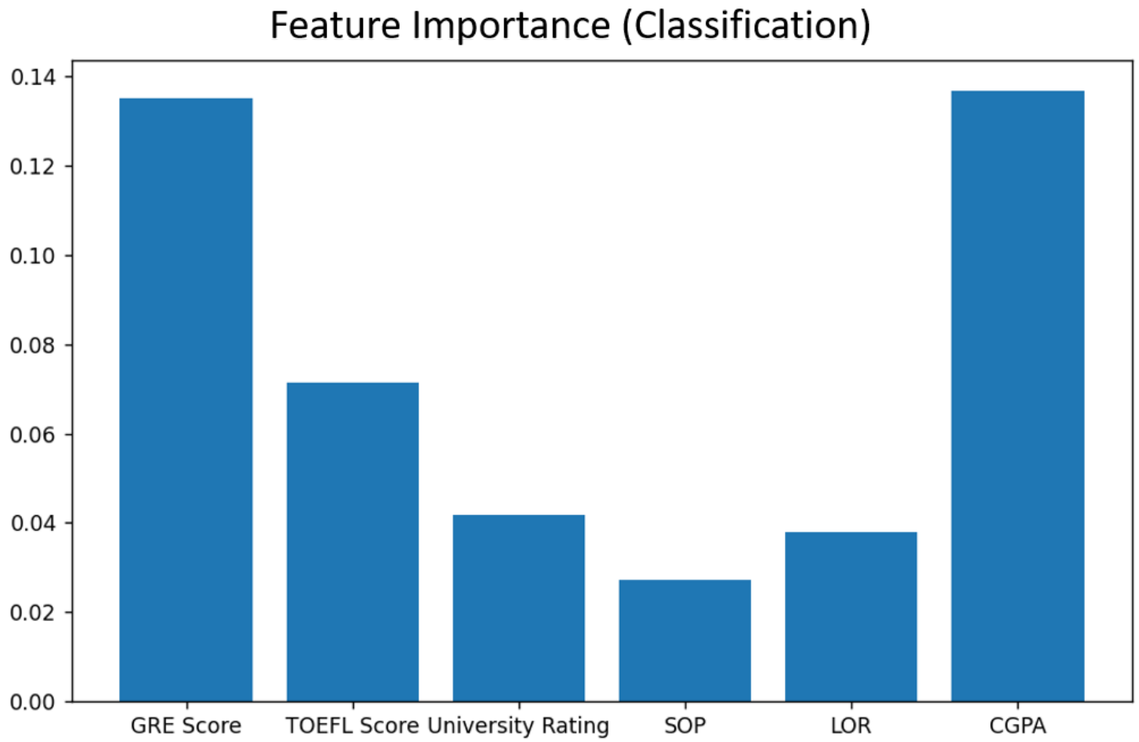
```
(400, 6, 2)
```

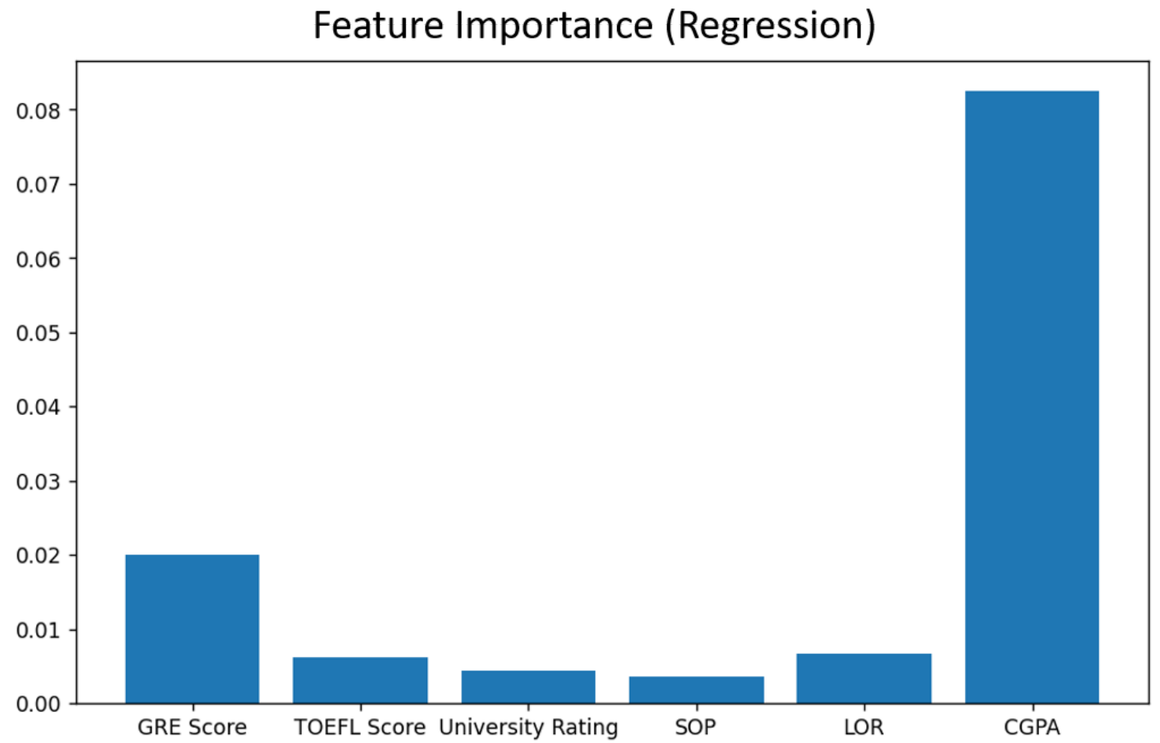
Select values of positive class

```
positive_values = shap_values_class[:,:,1]
```

Feature importance

```
mean_shap_values_class = np.abs(shap_values_class[:, :, 1]).mean(axis=0)
mean_shap_values_reg = np.abs(shap_values_reg).mean(axis=0)
plt.bar(X_train.columns, mean_shap_values_class)
plt.bar(X_train.columns, mean_shap_values_reg)
```





Let's practice!

EXPLAINABLE AI IN PYTHON

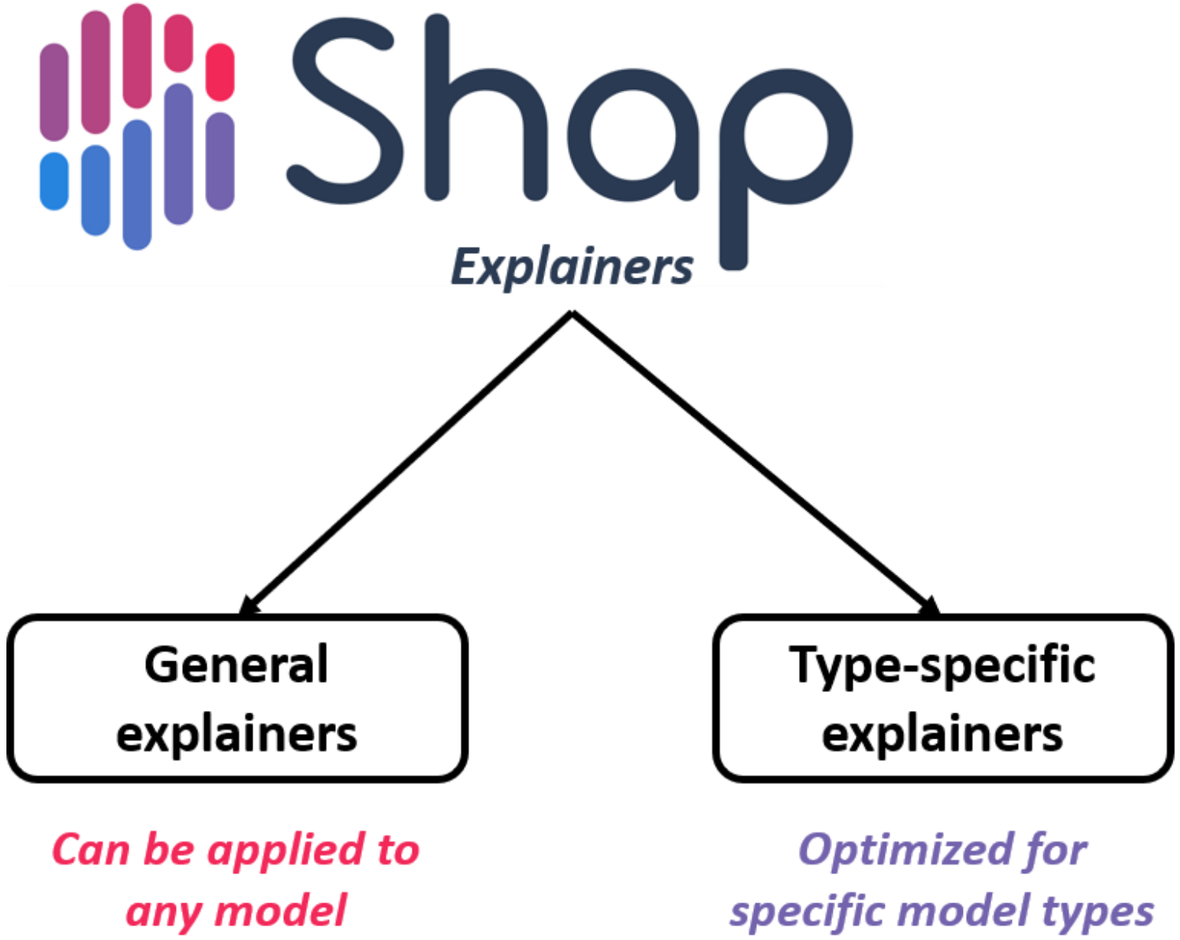
SHAP kernel explainer

EXPLAINABLE AI IN PYTHON

Fouad Trad
Machine Learning Engineer

SHAP kernel explainer

- Derives SHAP values for any model
  - K-nearest neighbors
  - Neural networks
  - Tree-based models
- Slower than type-specific explainers



Heart disease

age	sex	chest_pain_type	blood_pressure	ecg_results	thalassemia	target
52	1	0	125	1	3	0
53	1	0	140	0	3	0
70	1	0	145	1	3	0
61	1	0	148	1	3	0
62	0	0	138	1	2	0

`mlp_clf` : multilayer perceptron predicting risk of heart disease

Insurance charges

age	gender	bmi	children	smoker	charges
19	0	27.900	0	1	16884.92
18	1	33.770	1	0	1725.55
28	1	33.000	3	0	4449.46
33	1	22.705	0	0	21984.47
32	1	28.880	0	0	3866.85

`mlp_reg` : multilayer perceptron predicting insurance charges

Creating kernel explainers

MLPRegressor

```
import shap

explainer = shap.KernelExplainer(
  # Model's prediction function,
  # Representative summary of dataset
)
```

MLPClassifier

```
import shap

explainer = shap.KernelExplainer(
  # Model's prediction function,
  # Representative summary of dataset
)
```

Creating kernel explainers

MLPRegressor

```
import shap

explainer = shap.KernelExplainer(
  mlp_reg.predict,
  # Representative summary of dataset
)
```

MLPClassifier

```
import shap

explainer = shap.KernelExplainer(
  mlp_clf.predict_proba,
  # Representative summary of dataset
)
```

Creating kernel explainers

MLPRegressor

```
import shap

explainer = shap.KernelExplainer(
  mlp_reg.predict,
  shap.kmeans(X, 10)
)

shap_values_reg = explainer.shap_values(X)
```

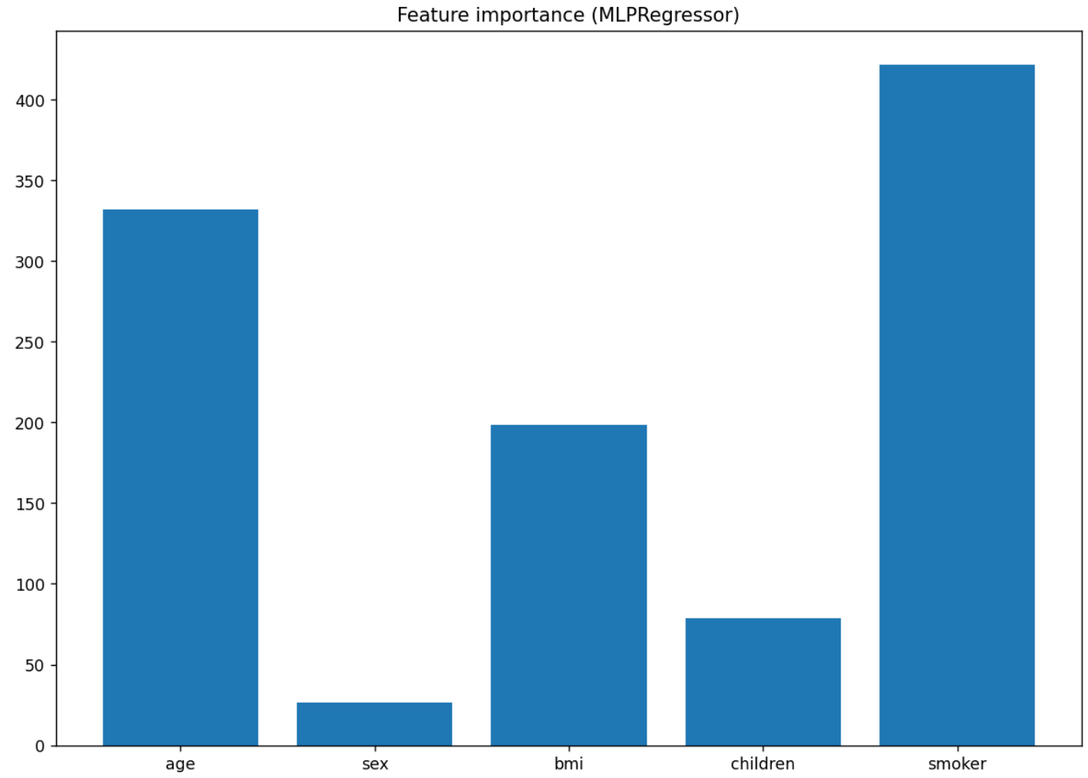
MLPClassifier

```
import shap

explainer = shap.KernelExplainer(
  mlp_clf.predict_proba,
  shap.kmeans(X, 10)
)

shap_values_cls = explainer.shap_values(X)
```

Feature importance

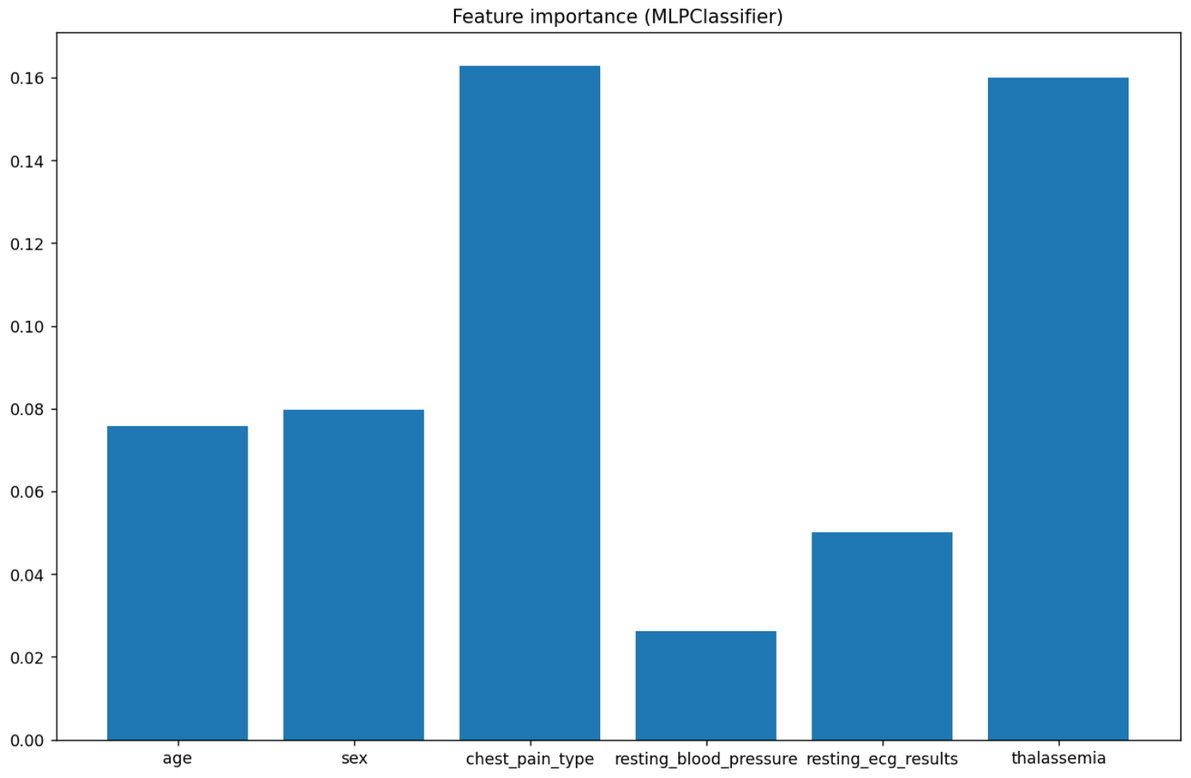
MLPRegressor

```
mean_reg = np.abs(shap_values_reg).mean(axis=0)
plt.bar(X.columns, mean_reg)
```



MLPClassifier

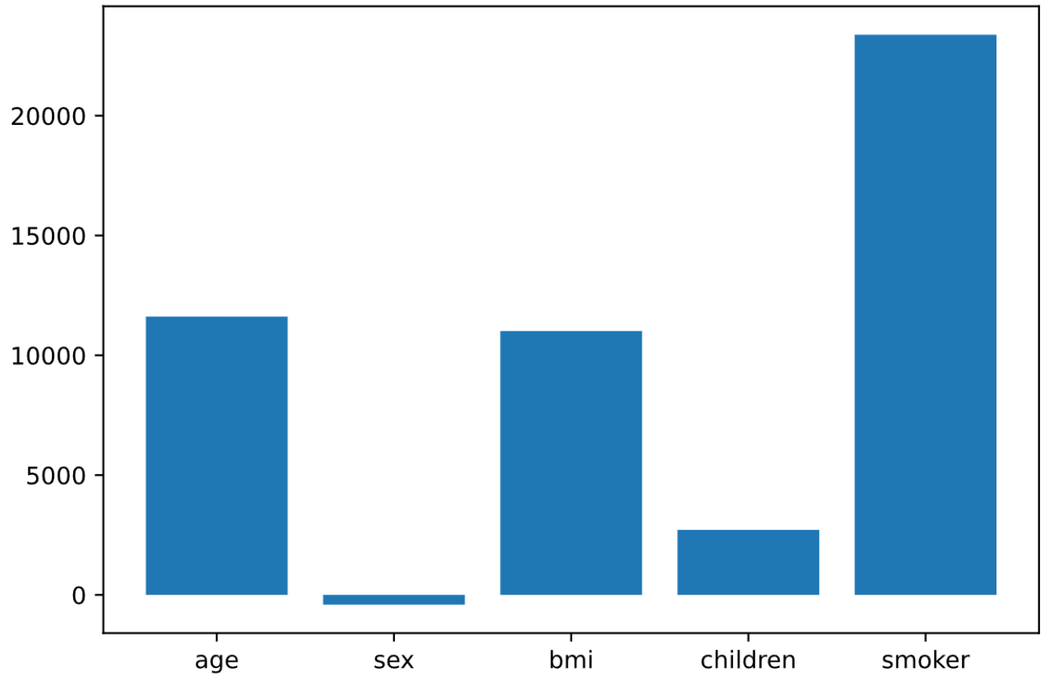
```
mean_cls = np.abs(shap_values_cls[:,:,1]).mean(axis=0)
plt.bar(X.columns, mean_cls)
```



Comparing with model-specific approaches

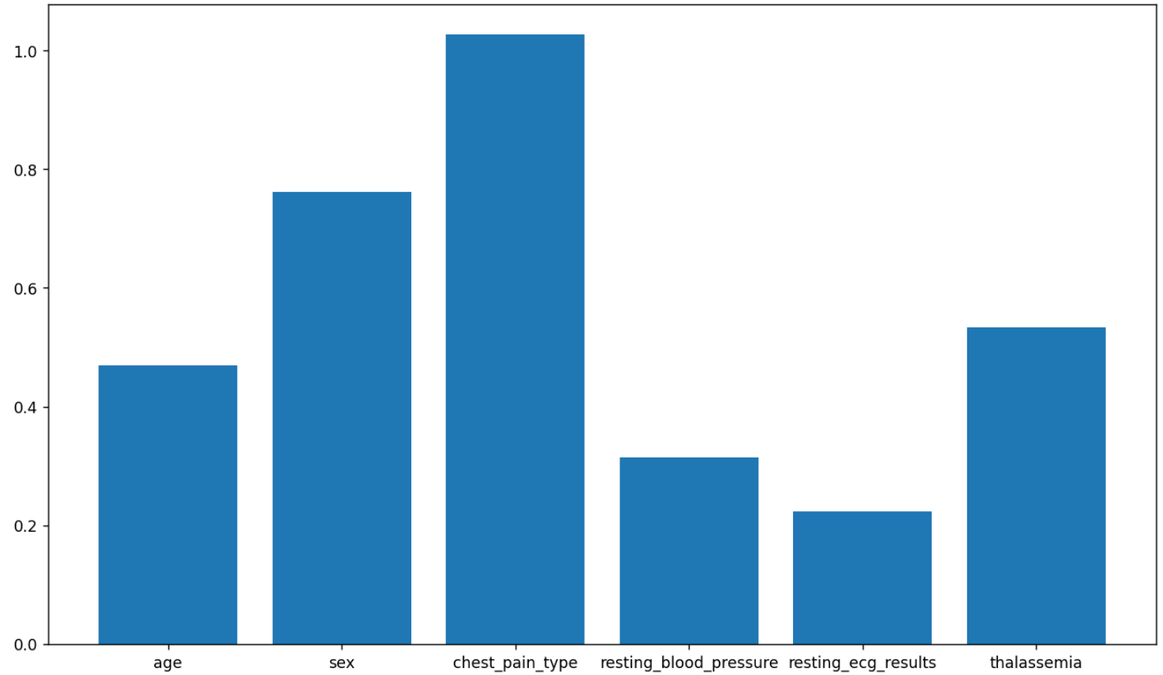
Linear regression

```
plt.bar(X.columns, np.abs(lin_reg.coef_))
```



Logistic regression

```
plt.bar(X.columns, np.abs(log_reg.coef_[0]))
```



Let's practice!

EXPLAINABLE AI IN PYTHON

Visualizing SHAP explainability

EXPLAINABLE AI IN PYTHON

Fouad Trad
Machine Learning Engineer

datacamp

Dataset

age	gender	bmi	children	smoker	charges
19	0	27.900	0	1	16884.92
18	1	33.770	1	0	1725.55
28	1	33.000	3	0	4449.46
33	1	22.705	0	0	21984.47
32	1	28.880	0	0	3866.85

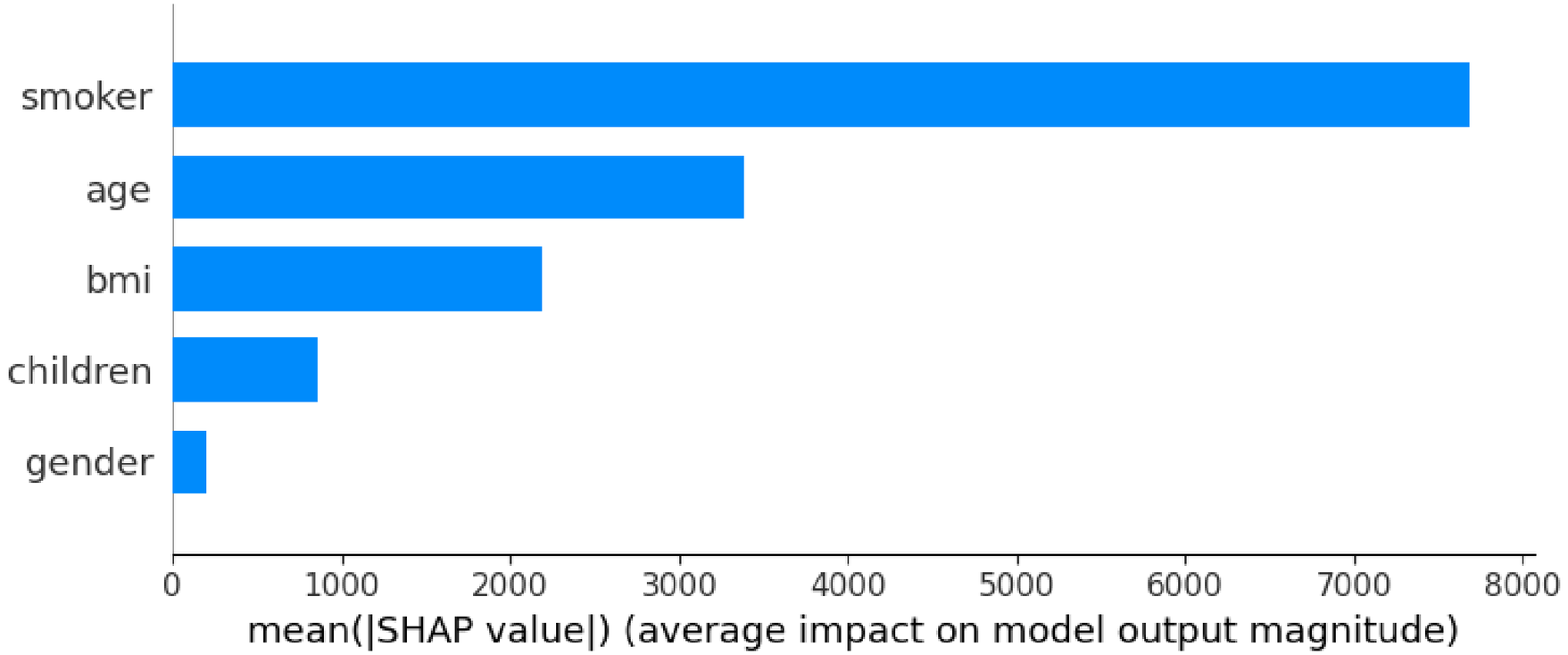
`model` : random forest regressor to predict charges

```
import shap
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
```

Feature importance plot

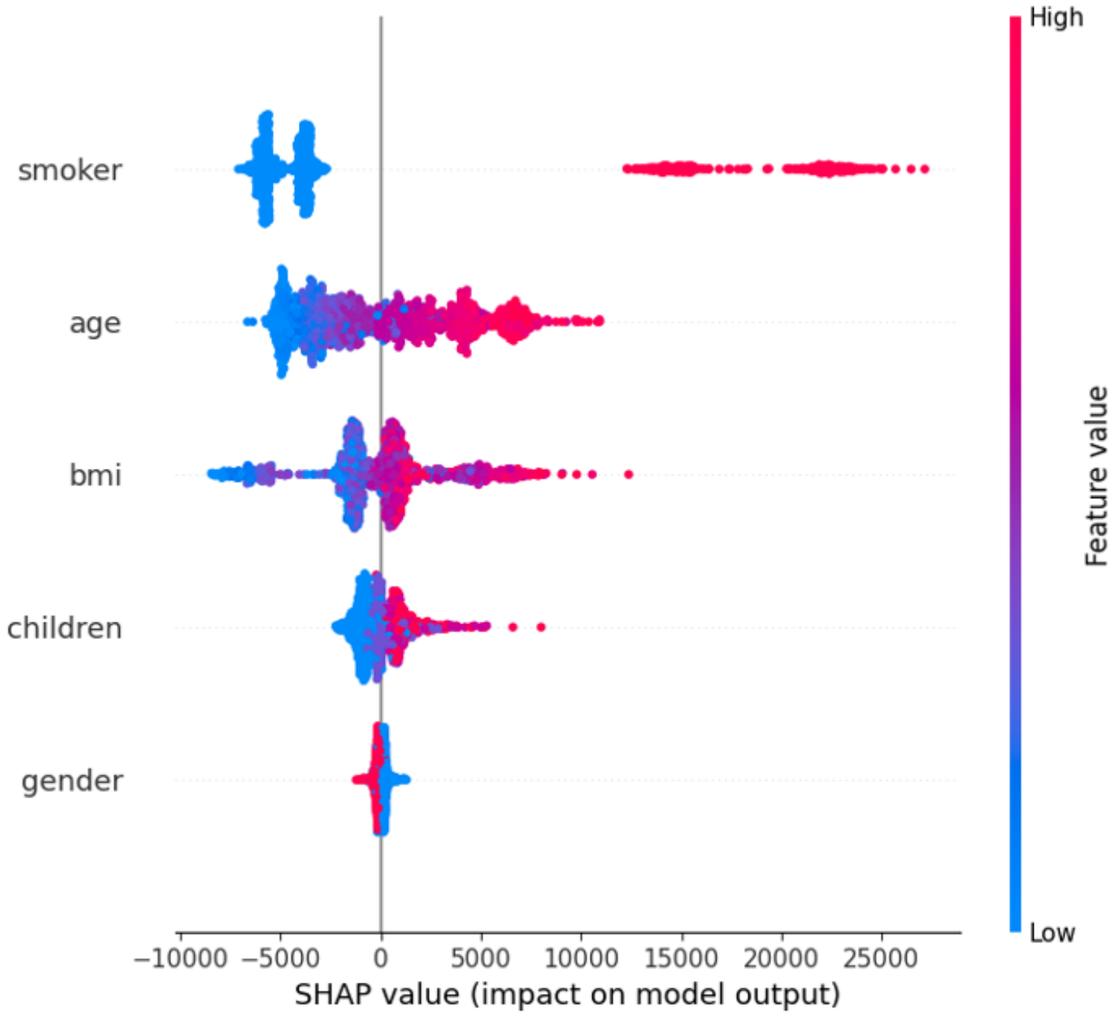
- Shows contribution of each feature on model output

```
shap.summary_plot(shap_values, X, plot_type="bar")
```



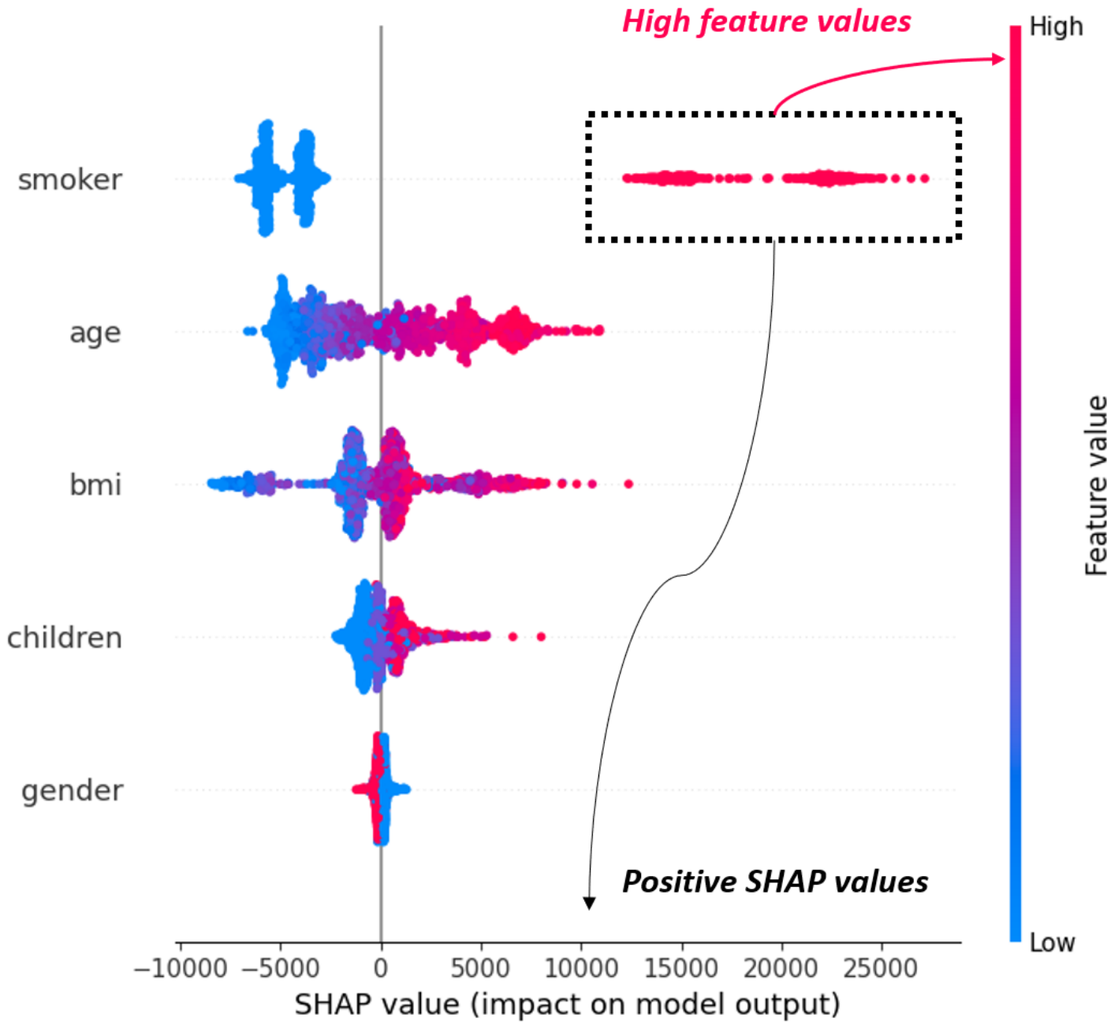
Beeswarm plot

- Shows SHAP values distribution
- Highlights **direction** and **magnitude** of each feature on prediction
  - Red color → high feature value
  - Blue color → low feature value
  - SHAP value > 0 → increases outcome
  - SHAP value < 0 → decreases outcome



Beeswarm plot

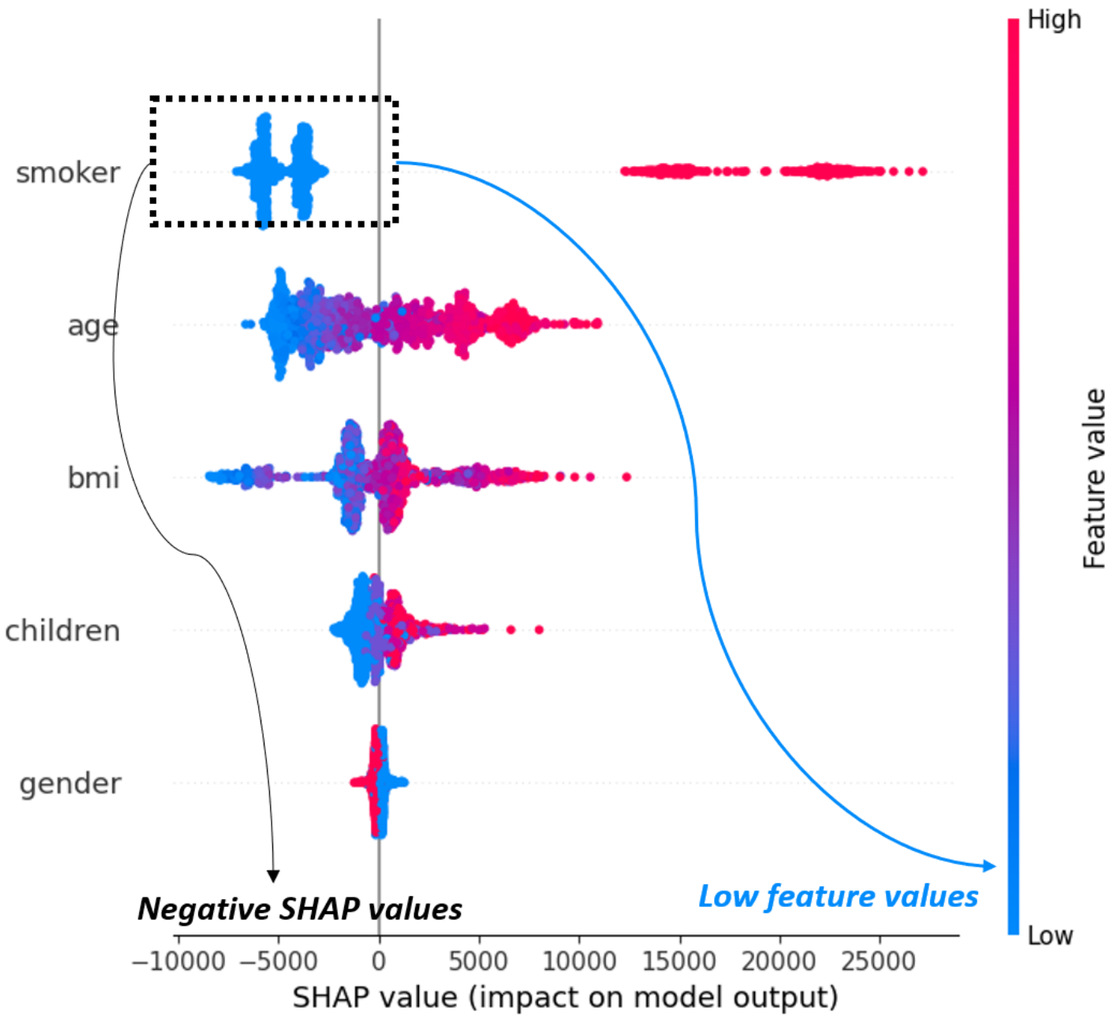
- Shows SHAP values distribution
- Highlights **direction** and **magnitude** of each feature on prediction
  - Red color → high feature value
  - Blue color → low feature value
  - SHAP value > 0 → increases outcome
  - SHAP value < 0 → decreases outcome



Beeswarm plot

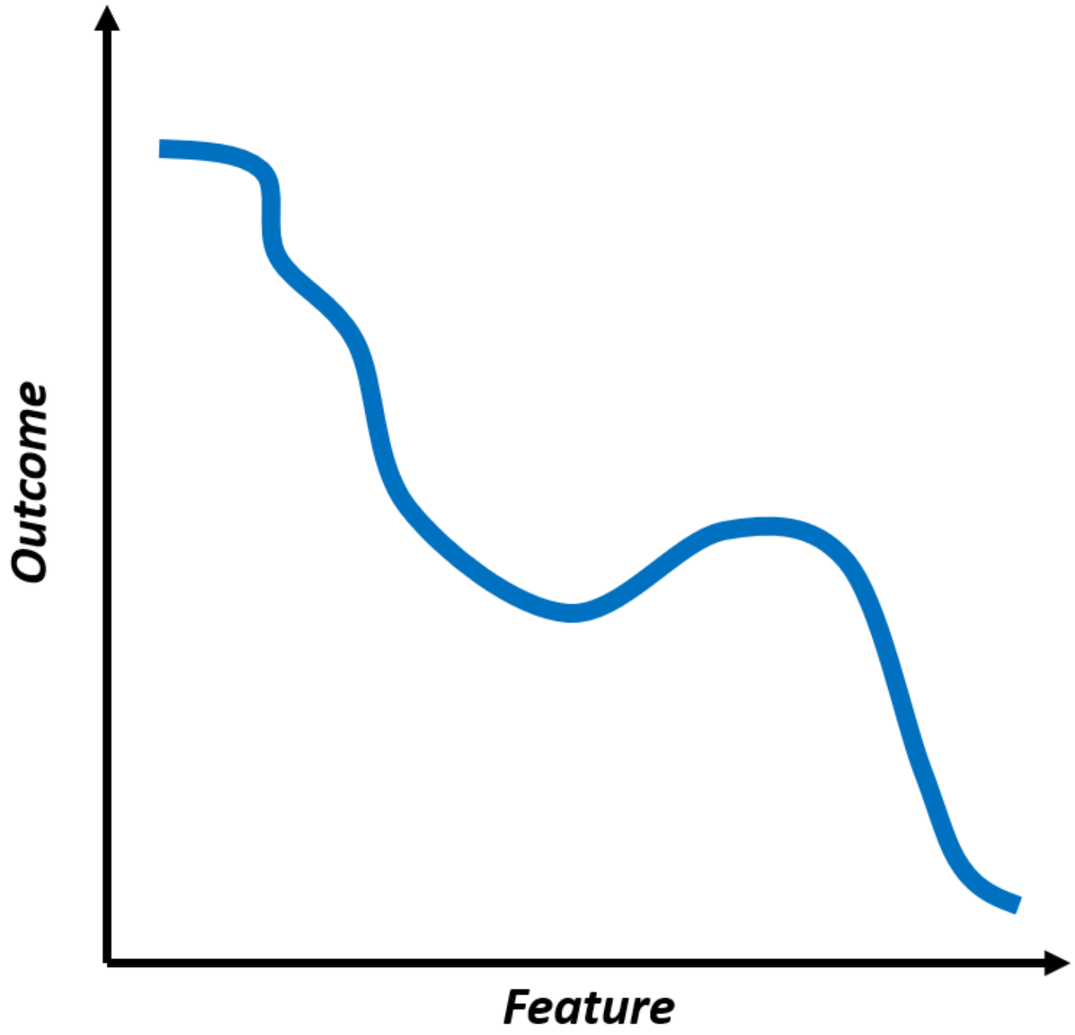
- Shows SHAP values distribution
- Highlights **direction** and **magnitude** of each feature on prediction
  - Red color → high feature value
  - Blue color → low feature value
  - SHAP value > 0 → increases outcome
  - SHAP value < 0 → decreases outcome

```
shap.summary_plot(shap_values, X,
                  plot_type="dot")
```



Partial dependence plot

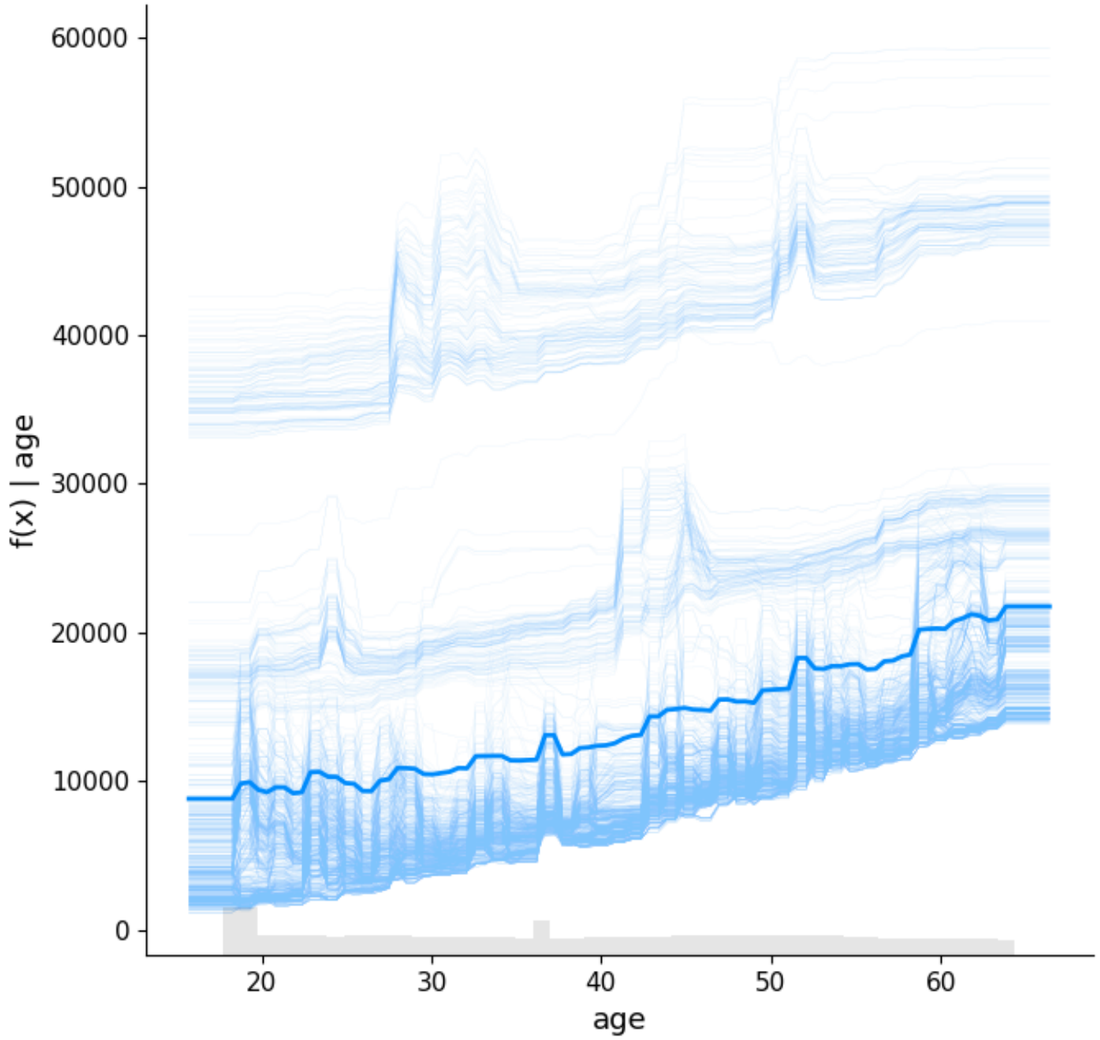
- Shows relationship between **feature** and **predicted outcome**
- Shows feature's impact across its range
- Verifies if relationship is as expected



Partial dependence plot

- For each sample:
  - Vary value of selected feature
  - Hold other features constant
  - Predict outcome
- Average results from all samples

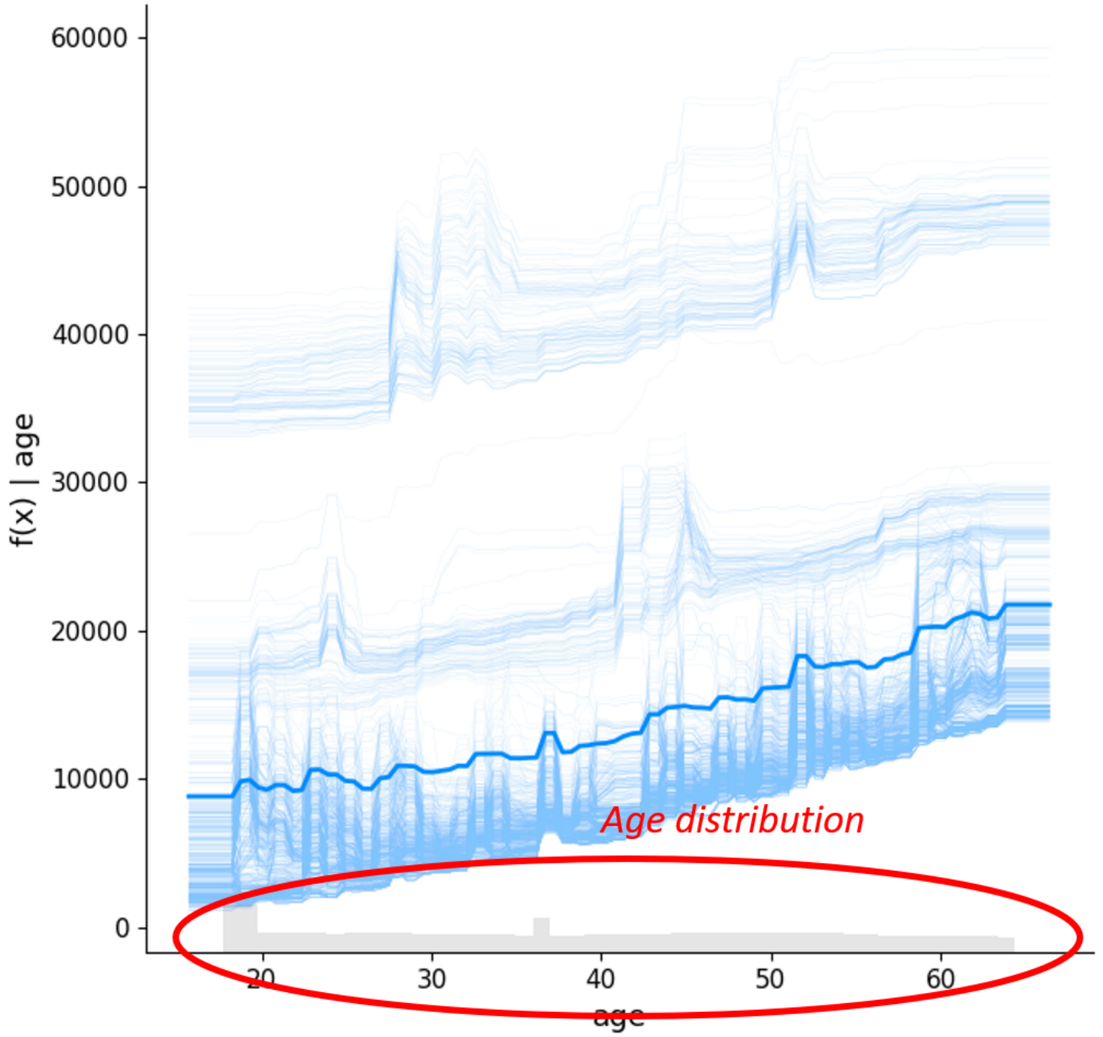
```
shap.partial_dependence_plot("age",
                             model.predict,
                             X)
```



Partial dependence plot

- For each sample:
  - Vary value of selected feature
  - Hold other features constant
  - Predict outcome
- Average results from all samples

```
shap.partial_dependence_plot("age",
                             model.predict,
                             X)
```



Let's practice!

EXPLAINABLE AI IN PYTHON