19 - Features Importance

Data Science Practicum 2021/22, Lesson 19

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Feature Importance

- Not all features are equally important
  - Feature Importance scores
    - Better understanding of the data.
    - Better understanding of the model.
    - Reducing the number of input features.

Coeficcients

- Linear ML models are a weighted sum of fetures
  - Weights -> importance
  - Data should be normalized
  - Linear regression, Logistic regression

Coeficcients

- Linear ML models are a weighted sum of fetures
  - Weights -> importance
  - Data should be normalized
  - Linear regression, Logistic regression

```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
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 = LinearRegression()
# fit the model
model.fit(X, y)
# get importance
importance = model.coef_
# 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()
```

```
Feature: 0, Score: -0.00000
Feature: 1, Score: 12.44483
Feature: 2, Score: 0.00000
Feature: 3, Score: -0.00000
Feature: 4, Score: 93.32225
Feature: 5, Score: 86.50811
Feature: 6, Score: 26.74607
Feature: 7, Score: 3.28535
Feature: 8, Score: -0.00000
Feature: 9, Score: 0.00000
```

- the model found the five important features and marked all other features with a zero coefficient, essentially removing them from the model.

Exercise

- Load the iris dataset
- Plot the histogram of the four features
- Use Logistic regression and print the features' importance `importance = model.coef_[0]`
- Z-Normalize the features (mean=0, std=1)
- Plot the histogram of the four features
- Use Logistic regression and print the features' importance

Exercise

- Load the iris dataset
- Plot the histogram of the four features
- Use Logistic regression and print the features' importance `importance = model.coef_[0]`
- Z-Normalize the features (mean=0, std=1)
- Plot the histogram of the four features
- Use Logistic regression and print the features' importance

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
pyplot.hist(X[:,0])
pyplot.hist(X[:,1])
pyplot.hist(X[:,2])
pyplot.hist(X[:,3])
pyplot.show()

# fit-1
model = LogisticRegression()
model.fit(X, y)
importance = model.coef_[0]
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: -0.41813
Feature: 1, Score: 0.96633
Feature: 2, Score: -2.52103
Feature: 3, Score: -1.08410
```

```
#Z normalization
X = preprocessing.scale(X)
pyplot.hist(X[:,0])
pyplot.hist(X[:,1])
pyplot.hist(X[:,2])
pyplot.hist(X[:,3])
pyplot.show()

# fit-2
model = LogisticRegression()
model.fit(X, y)
importance = model.coef_[0]
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: -1.07404
Feature: 1, Score: 1.16006
Feature: 2, Score: -1.93063
Feature: 3, Score: -1.81169
```

Decision Tree Feature Importance

- Decision Tree-based models offer importance score
  - Decision Tree
  - Random Forest

Decision Tree Feature Importance

- Decision Tree-based models offer importance score
  - Decision Tree
  - Random Forest

```
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
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))
```

```
Feature: 0, Score: 0.00280
Feature: 1, Score: 0.00424
Feature: 2, Score: 0.00179
Feature: 3, Score: 0.00243
Feature: 4, Score: 0.51731
Feature: 5, Score: 0.43768
Feature: 6, Score: 0.02690
Feature: 7, Score: 0.00306
Feature: 8, Score: 0.00274
Feature: 9, Score: 0.00104
```

Exercise

- Iris dataset
- DecisionTreeClassifier
- Print features importance

Exercise

- Iris dataset
- DecisionTreeClassifier
- Print features importance

```
iris = datasets.load_iris()
X = iris.data
y = iris.target

model = DecisionTreeClassifier()
model.fit(X, y)
importance = model.feature_importances_
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

Permutation Feature Importance

- Model agnostic
- Algorithm:
  - Take a model that was fit to the training dataset
  - Estimate the predictive performance of the model on an independent dataset (e.g., validation dataset) and record it as the baseline performance
  - For each feature i:
    - randomly permute feature column i in the original dataset
    - record the predictive performance of the model on the dataset with the permuted column
    - compute the feature importance as the difference between the baseline performance (step 2) and the performance on the permuted dataset

Permutation Feature Importance

```python
# permutation feature importance with knn for regression
from sklearn.datasets import make_regression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.inspection import permutation_importance
from matplotlib import pyplot

X, y = make_regression(n_samples=1000, n_features=10,
    n_informative=5, random_state=1)
model = KNeighborsRegressor()
model.fit(X, y)

# perform permutation importance
results = permutation_importance(model, X, y,
    scoring='neg_mean_squared_error')
# get importance
importance = results.importances_mean
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 148.10397
Feature: 1, Score: 339.66791
Feature: 2, Score: 144.91106
Feature: 3, Score: 57.40640
Feature: 4, Score: 9704.56218
Feature: 5, Score: 7821.59872
Feature: 6, Score: 884.23434
Feature: 7, Score: 114.10732
Feature: 8, Score: 141.39661
Feature: 9, Score: 119.15848
```

Exercise

- Classification dataset (1000,10,5)
- Train test splitting 0.33, LogisticRegression, Accuracy Score
- Calculate the accuracy score with the top 1,2,3,4,5 features

Exercise

- Classification dataset (1000,10,5)
- Train test splitting 0.33, LogisticRegression, Accuracy Score
- Calculate the accuracy score with the top 1,2,3,4,5 features

```
X, y = make_classification(n_samples=1000, n_features=10,
    n_informative=5, random_state=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.33, random_state=1)
model = LogisticRegression()
model.fit(X_train, y_train)
results = permutation_importance(model, X, y,
    scoring='neg_mean_squared_error')
importance = results.importances_mean
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
model = LogisticRegression()
model.fit(X_train, y_train)
y_test_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_test_pred)
print('Accuracy: %.2f' % (accuracy*100))
#
X_train = X_train[:,[5]]
X_test = X_test[:,[5]]
model = LogisticRegression()
model.fit(X_train, y_train)
y_test_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_test_pred)
print('Accuracy: %.2f' % (accuracy*100))
```

```
Feature: 0, Score: 0.00660
Feature: 1, Score: 0.08380
Feature: 2, Score: 0.05100
Feature: 3, Score: 0.01460
Feature: 4, Score: 0.00120
Feature: 5, Score: 0.13140
Feature: 6, Score: 0.00240
Feature: 7, Score: -0.00040
Feature: 8, Score: 0.01340
Feature: 9, Score: 0.02580
Accuracy: 83.64
Accuracy: 46.97
```

Table of Contents

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

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