Алгоритмы машинного обучения

Стандартизация признаков

In
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() # создаём объект класса scaler
scaler.fit(X) # обучаем стандартизатор
X_sc = scaler.transform(X) # преобразуем набор данных
```

Вычисление корреляционной матрицы датафрейма

In
```
cm = df.corr()
```

Инициализация моделей линейной регрессии

In
```
# Инициализация моделей линейной регрессии
from sklearn import linear_model
linear_regression = linear_model.LinearRegression()
# линейная регрессия со встроенной L1 регуляризацией
lasso = linear_model.Lasso()
# линейная регрессия со встроенной L2 регуляризацией
ridge = linear_model.Ridge()

# обучение модели
linear_regression.fit(X_train, y_train)

# получение предсказаний
y_pred = linear_regression.predict(X_val)

# получение коэффициентов линейной регрессии
print(model.coef_)
# получение нулевого коэффициента регрессии
print(model.intercept_)
```

Получение значения MAE обученной модели

In
```
# Аналогично вычисляются MSE (mean_squared_error) и R2 (r2_score)
from sklearn.metrics import mean_absolute_error

print('MAE:', mean_absolute_error(y_true, y_pred))
```

Инициализация модели логистической регрессии (алгоритма бинарной классификации)

In
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

# получение вероятностей принадлежности классам
y_probas = model.predict_proba(X_test)
```

Получение матрицы ошибок для классификации

In
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_true,y_pred)
tn, fp, fn, tp = cm.ravel() # "выпрямляем" матрицу, чтобы вытащить нужные значения
```

Рассчёт метрик качества классификации

In
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
                            f1_score, roc_auc_score
acc = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
roc_auc = roc_auc_score(y_true, probabilities[:,1])
```

Обучение модели решающего дерева для классификации

In
```
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
y_pred = tree_model.fit(X_test)

# Визуализация обученного дерева решений

plt.figure(figsize = (20,15)) # задайте размер фигуры, чтобы получить крупное изображение
plot_tree(tree_model, filled=True, feature_names = X_train.columns,
          class_names = ['not fault', 'fault'])
plt.show()
```

Обучение модели случайного леса для регрессии

In
```
from sklearn.ensemble import RandomForestRegressor

# n_estimators — количество деревьев решений
rf_model = RandomForestRegressor(n_estimators = 100)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_val)
```

Обучение модели градиентного бустинга для регрессии

In
```
from sklearn.ensemble import GradientBoostingRegressor

# n_estimators — количество простых моделей
gb_model = GradientBoostingRegressor(n_estimators = 100)
gb_model.fit(X_train, y_train)
y_pred = gb_model.predict(X_val)
```

Кластеризация методом KMeans

In
```
from sklearn.cluster import KMeans

# обязательная стандартизация данных перед работой с алгоритмами
sc = StandardScaler()
X_sc = sc.fit_transform(X)

km = KMeans(n_clusters = 5) # задаём число кластеров, равное 5
labels = km.fit_predict(X_sc) # применяем алгоритм к данным и формируем вектор кластеров
```

Построение иерархической кластеризации

In
```
from scipy.cluster.hierarchy import dendrogram, linkage

# обязательная стандартизация данных перед работой с алгоритмами
sc = StandardScaler()
X_sc = sc.fit_transform(X)

linked = linkage(X_sc, method = 'ward')

plt.figure(figsize=(15, 10))
dendrogram(linked, orientation='top')
plt.title('Hierarchial clustering for GYM')
plt.show()
```

Вычисление метрики силуэта для кластеризации

In
```
from sklearn.metrics import silhouette_score

silhouette_score(x_sc, labels)
```

Словарь

Глобальный минимум функции
минимум функции среди всех возможных значений функции

Локальный минимум функции
минимум функции среди значений функции в некоторой окресности данной точки

Градиентный спуск
это особый метод поиска локального минимума функции

Градиент
вектор, состоящий из частных производных

Гиперпараметры модели
параметры, значения которых задается до начала обучения модели и не изменяется в процессе обучения. У модели может не быть гиперпараметров.

Нормализация
перевод значений признаков в диапазон от 0 до 1

Стандартизация
перевод значений признаков таким образом, чтобы они были случайными величинами из **стандартного нормального распределения** с математическим ожиданием 0 и дисперсией 1

Мультиколлинеарность
тесная корреляционная взаимосвязь между несколькими признаками

Регуляризация
метод добавления некоторых дополнительных ограничений к условию с целью предотвратить переобучение. Чаще всего эта информация имеет вид штрафа за сложность модели