Resumen del capítulo: Detección de anomalías

Anomalías

Las a**nomalías,** o o**utliers en inglés,** son observaciones con propiedades atípicas (es decir, que se desvían de la tendencia normal). Los valores atípicos indican un problema en los datos o que algo está fuera de lo normal. Un ejemplo de esto podría ser una transacción bancaria sospechosa: John de St. Louis, quien suele contentarse con una o dos visitas a un restaurante a la semana, de repente se compra un smartphone muy costoso en Hanói. Aquí se pueden observar o predecir algunas anomalías. Por ejemplo, podemos estar atentos a las anomalías meteorológicas: si los datos apuntan a un invierno más cálido de lo habitual, los clientes no se molestarán en comprar ropa de abrigo adicional. ("¡Mejor me compro un abrigo ligero con estilo, o esos zapatos que siempre he querido!").

Dado que los valores atípicos suelen ser impredecibles y poco comunes, durante el entrenamiento aparecen pocas o ninguna anomalía.

Diagrama de caja

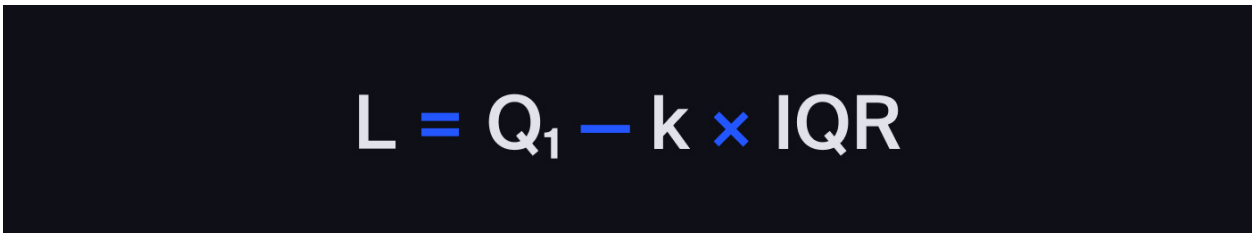
Imaginemos los valores de las características como un conjunto de números. Necesitamos encontrar los pocos números que difieren mucho de los demás. Para hacer esto, vamos a compararlos con la mediana en un **diagrama de caja**, que también se llama **gráfico de caja y bigotes** debido a las líneas que se extienden desde las cajas como bigotes.

Los límites superior e inferior de la caja marcan los cuartiles primero y tercero (75% y 25% de los valores). La mediana se ubica en el centro (50% de los valores). Los "bigotes" se extienden hacia arriba y hacia abajo desde los bordes de la caja hasta una distancia de 1.5 **intervalos intercuartílicos** (**IQR**) e indican la variabilidad fuera de los cuartiles inferior y superior. Los valores atípicos se muestran fuera de los límites de los bigotes (el mínimo y el máximo).

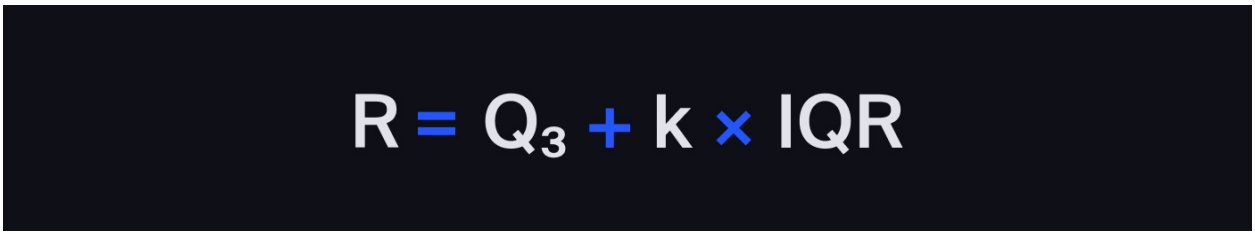
El IQR se calcula de esta manera:

$$IQR = Q_3 - Q_1$$

La fórmula para la valla interior inferior de la caja (aquí, **k** es el coeficiente para **intervalos intercuartílicos**, normalmente establecido en 1.5) es:



La fórmula para la valla exterior superior:



Cuanto mayor sea el coeficiente k, menos valores atípicos habrá.

El diagrama muestra información sobre todos los valores atípicos. Se almacena en la lista `"fliers"` dentro de la instancia de `boxplot`. Podemos ver cuántos valores atípicos hay llamando a la función `get_data()`. Los valores necesarios estarán separados por índices.

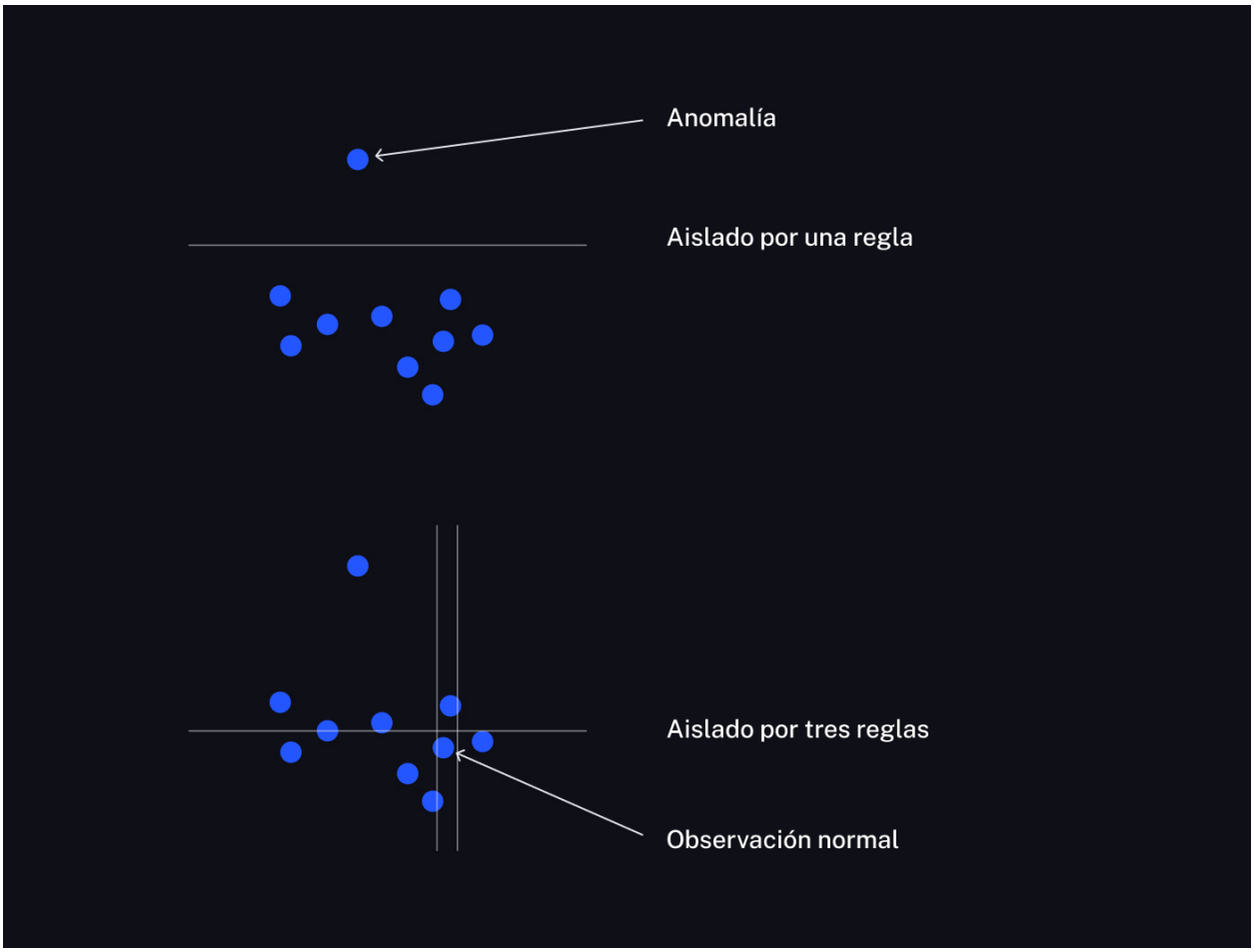
```
boxplot = plt.boxplot(df['column'].values)

# Lista de valores atípicos
outliers = list(boxplot["fliers"][0].get_data()[1])
print("Valores atípicos: ", len(outliers))
```

Bosque de aislamiento

Este es un método de ensamble. Sus cálculos se basan en las estimaciones promediadas de varios árboles de solución. Los nodos del árbol contienen las **reglas de decisión** que asignan cada observación a una rama específica.

Este método se basa en el hecho de que las anomalías se pueden aislar del resto mediante un pequeño número de reglas de decisión.



Un árbol de aislamiento se construye como un árbol de decisión, pero las reglas de decisión para este se eligen al azar. Las observaciones situadas a poca profundidad se pueden aislar fácilmente y se consideran anómalas, mientras que el resto se consideran normales. Las estimaciones de las anomalías se recopilan de todos los árboles y se promedian.

Veamos cómo se puede entrenar un bosque aislado en la librería sklearn. Para ello, importaremos la clase `IsolationForest()` desde el módulo `sklearn.ensemble`:

```
from sklearn.ensemble import IsolationForest
```

Vamos a crear un modelo y registrar la cantidad de árboles en el parámetro `n_estimators`. Cuantos más árboles haya, más precisos serán los resultados:

```
isolation_forest = IsolationForest(n_estimators=100)
```

Para encontrar anomalías en datos unidimensionales, convierte `df['Sales']` en una matriz bidimensional:

```
sales = df['column'].values.reshape(-1, 1)
```

Seleccionar anomalías por una característica no te dará una representación precisa del conjunto de datos completo. Un bosque de aislamiento detecta valores atípicos con base en varias características.

```
data = df[['column1', 'column2']]
```

El entrenamiento adicional del modelo es el mismo tanto para los datos unidimensionales como para los multidimensionales. Ahora vamos a entrenar el modelo con los datos de ventas y beneficios utilizando `fit()`:

```
isolation_forest.fit(data)
```

Al llamar a `decision_function()` veremos exactamente cómo evaluó el modelo las observaciones:

```
anomaly_scores = isolation_forest.decision_function(data)
```

Las estimaciones de anomalías varían entre -0.5 y 0.5. Una estimación más baja indica una mayor probabilidad de que la observación sea un valor atípico.

Para calcular el número de anomalías, llamemos al método `predict()`, que clasifica las observaciones y distingue las normales de las atípicas. Si la clase de observación es 1,

la observación es normal, pero si es -1, es un valor atípico.

```
estimator = isolation_forest.predict(data)
```

Si las estimaciones de anomalías no son necesarias, puedes entrenar el modelo y obtener la clasificación usando `fit_predict()`:

```
estimator = isolation_forest.fit_predict(data)
```

Método de detección de anomalías basado en KNN

Hay otra forma de encontrar anomalías en datos multidimensionales, que es el **algoritmo de k-vecinos más cercanos** (KNN). Este algoritmo funciona así: considera cada observación del conjunto de datos como un vector y busca anomalías en el espacio multidimensional. Cuanto más lejos esté una observación de sus vecinos, mayor será la probabilidad de que sea un valor atípico.

La clase `KNN()` se encuentra en la librería **PyOD** (kit de herramientas de Python para detectar objetos periféricos). Impórtala desde el módulo `pyod.models.knn`:

```
from pyod.models.knn import KNN
```

Vamos a entrenar el modelo en el conjunto llamando al método `fit()`:

```
model = KNN()
model.fit(data)
```

Una vez que se complete el entrenamiento, puedes continuar con la búsqueda de anomalías en el conjunto de datos. Vamos a llamar al método `predict()`:

```
predictions = model.predict(data)
```

El método `predict()` devolverá una lista donde "1" indica una anomalía y "0" significa que la observación es parte de la tendencia general.