Thursday, November 28, 2024 6:58 PM

Мапирање:

1. **Трансформација на категориски податоци**

За конверзија на категориски податоци (на пример, `Yes/No` или `Male/Female`) во бројки кои може да ги користат алгоритмите за машинско учење:

python Copy code

```
mapping = {'Yes': 1, 'No': 0}
df['Category'] = df['Category'].map(mapping)
```

3. **Форматирање податоци**

Пример: менување на датумите во специфичен формат.

python Copy code

```
df['Formatted Date'] = df['Date'].map(lambda x: x.strftime('%Y-%m-%d'))
```

1. **Користење на** `.replace()`

python Copy code

```
default_values = {'A': 10, 'B': 20}
df['Column'] = df['Column'].replace(default_values)
```

7. **Обработување текстуални податоци**

За трансформирање или чистење текст:

python Copy code

```
df['Cleaned Text'] = df['Text'].map(lambda x: x.lower().strip())
```

Со дефинирање на функција

```python
def clean_text(x):
    return x.lower().strip()

df['Cleaned Text'] = df['Text'].map(clean_text)
```

Missing values:

- **Отстранување на редови/колони** со премногу недостасувачки вредности.
- **Пополнување** со статистички мерки (средина, медијана, мода) или предвидување.

- Ако фалат многу податоци, ги отстрануваме
- Ако фалат податоци во target колоната, ги отстрануваме редовите каде што фалат

1. `fillna` **(од Pandas):**

- **Опис:**
  - Се користи за да се пополнат недостасувачки вредности (`NaN`) во колони или редови на DataFrame или Series.
  - Корисникот директно наведува вредност (или функција, како `mean` или `median`) со која ќе се заменат празнините.

df['Age'] = df['Age'].fillna(df['Age'].median()) # Медијана
df['Salary'] = df['Salary'].fillna(df['Salary'].mean()) # Средна вредност

2. `SimpleImputer` **(од Scikit-learn):**

- **Опис:**
  - Ова е класа која обезбедува понапреден пристап за пополнување на недостасувачки вредности во склоп на предобработката на податоци.
  - Работи со цел за обука и тестирање податоци преку два чекора:
    - `fit` (учење на вредноста за импутирање, како медијана, на податоци од обука).
    - `transform` (импутирање на вредноста во нови податоци).

df = pd.DataFrame({'Age': [25, 30, None, 35], 'Salary': [50000, 55000, 60000, None]})

# Иницијализација на SimpleImputer за медијана
imputer = SimpleImputer(strategy='median')

Импутирање на податоците
df[['Age', 'Salary']] = imputer.fit_transform(df[['Age', 'Salary']])

3. Dropping Unnecessary Columns (Отстранување на непотребни колони)

Зошто?

Колоните кои немаат значајна врска со целта можат да создадат шум.

```
df = df.drop(columns=['City'])
```

4. Encoding Categorical Data (Енкодирање на категоријални податоци)

- **Label Encoding** за категории со природен редослед.
- **One-Hot Encoding** за категории без природен редослед.

OneHotEncoding ги претвора категоријалните вредности во бинарни вектори.

```
df = pd.DataFrame({'Color': ['Red', 'Blue', 'Green']})
encoder = OneHotEncoder(sparse=False)
encoded = encoder.fit_transform(df[['Color']])
encoded_df = pd.DataFrame(encoded, columns=encoder.get_feature_names_out())
print(encoded_df)
```

5. Feature Scaling (Скалирање на податоците)

- **Min-Max Scaling**: Промена на вредностите во опсегот [0, 1].
- **Standardization**: Зачувување на нормалната распределба.

```
scaler = MinMaxScaler() df[['Age', 'Salary']] = scaler.fit_transform(df[['Age', 'Salary']]) print(df)
```

6. Splitting Data (Поделба на податоци)

```
X = df[['Age', 'Salary']]
y = [1, 0, 1]  # Целна променлива
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train, X_test, y_train, y_test)
```