Προ-επεξεργασία Δεδομένων

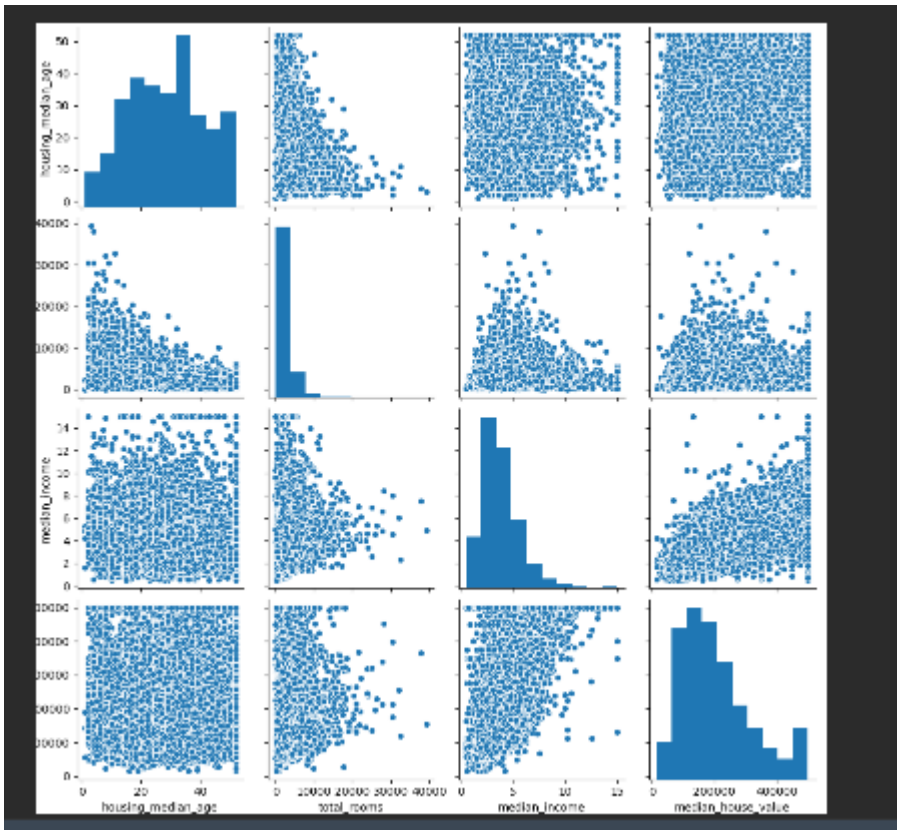
Οπτικοποίηση Δεδομένων

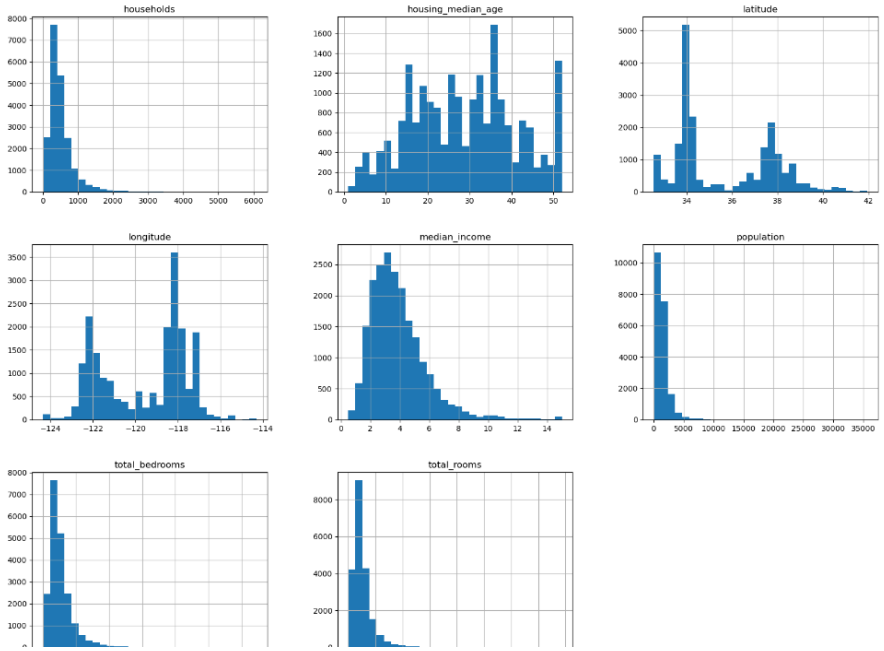
ΦΡΑΝΤΖΗΣ ΚΟΡΝΗΛΑΚΗΣ Π20095

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv("C:\\Users\\frantzis\\Desktop\\housing.csv")

# Identify the numerical and categorical columns
numerical_cols = ['longitude', 'latitude', 'housing_median_age',
'total_rooms',
                  'total_bedrooms', 'population', 'households',
'median_income']
categorical_cols = ['ocean_proximity']

# Preprocessing for numerical data
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')), # Fill missing
values with median
    ('scaler', StandardScaler()), # Scale numerical features
])

# Preprocessing for categorical data
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Create a copy of the data
data_copy = data.copy()

# Remove the target column from the data
data_copy.drop('median_house_value', axis=1, inplace=True)

# Apply the transformations to the data
data_preprocessed = preprocessor.fit_transform(data_copy)

# Convert the processed data back to a DataFrame
data_preprocessed = pd.DataFrame(data_preprocessed)

# Check the transformed data
print(data_preprocessed.head())
```

```python
# Histograms for each numerical variable
data[numerical_cols].hist(bins=30, figsize=(20,15))
plt.show()

# Pairplot for a subset of variables
subset_of_cols = ['housing_median_age', 'total_rooms',
'median_income', 'median_house_value']
sns.pairplot(data[subset_of_cols])
plt.show()
```

```
\Users\frantzis\anaconda3\python.exe C:\Users\frantzis\PycharmProjects\pythonProject52\
       0         1         2         3         4   ...   8    9    10   11   12
1.327835  1.052548  0.982143 -0.804819 -0.972476  ...  0.0  0.0  0.0  1.0  0.0
1.322844  1.043185 -0.607019  2.045890  1.357143  ...  0.0  0.0  0.0  1.0  0.0
1.332827  1.038503  1.856182 -0.535746 -0.827024  ...  0.0  0.0  0.0  1.0  0.0
1.337818  1.038503  1.856182 -0.624215 -0.719723  ...  0.0  0.0  0.0  1.0  0.0
1.337818  1.038503  1.856182 -0.462404 -0.612423  ...  0.0  0.0  0.0  1.0  0.0

rows x 13 columns]
```





Παλινδρόμηση Δεδομένων

Perceptron

[-1 1 1 ... 1 1 1]

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# Load the data
data = pd.read_csv("C:\\Users\\frantzis\\Desktop\\housing.csv")

# Identify the numerical and categorical columns
numerical_cols = ['longitude', 'latitude', 'housing_median_age',
'total_rooms',
                  'total_bedrooms', 'population', 'households',
'median_income']
categorical_cols = ['ocean_proximity']

# Fill missing values with the median of the respective columns
for col in numerical_cols:
    data[col] = data[col].fillna(data[col].median())

# Apply one-hot encoding to the categorical columns
encoder = OneHotEncoder(sparse=False)
encoded_categorical_cols =
pd.DataFrame(encoder.fit_transform(data[categorical_cols]))
```

```
# Drop the original categorical columns from the data
data = data.drop(categorical_cols, axis=1)

# Concatenate the original data with the encoded categorical columns
data = pd.concat([data, encoded_categorical_cols], axis=1)

# Define the threshold for binary classification
threshold = data['median_house_value'].median()
data['median_house_value'] = data['median_house_value'].apply(lambda
x: 1 if x > threshold else -1)

# Split the data into features and target
X = data.drop('median_house_value', axis=1).values
y = data['median_house_value'].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Check for infinite values and replace them if any
X_train = np.nan_to_num(X_train)
X_test = np.nan_to_num(X_test)

class Perceptron:
    def __init__(self, learning_rate=0.01, n_iters=1000):
        self.lr = learning_rate
        self.n_iters = n_iters
        self.activation_func = self._unit_step_func
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape

        # init parameters
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_predicted = self.activation_func(linear_output)

                # Perceptron update rule
                update = self.lr * (y[idx] - y_predicted)

                self.weights += update * x_i
                self.bias += update

    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        y_predicted = self.activation_func(linear_output)
        return y_predicted

    def _unit_step_func(self, x):
        return np.where(x>=0, 1, -1)

# Initialize and train the perceptron
p = Perceptron(learning_rate=0.01, n_iters=1000)
p.fit(X_train, y_train)
```

```
# Make predictions on the test data
predictions = p.predict(X_test)

# Print the predictions
print(predictions)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
data = pd.read_csv("C:\\Users\\frantzis\\Desktop\\housing.csv")

# Identify the numerical and categorical columns
numerical_cols = ['longitude', 'latitude', 'housing_median_age',
'total_rooms',
                  'total_bedrooms', 'population', 'households',
'median_income']
categorical_cols = ['ocean_proximity']

# Preprocessing for numerical data
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')), # Fill missing
values with median
    ('scaler', StandardScaler()), # Scale numerical features
])

# Preprocessing for categorical data
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Create a copy of the data
data_copy = data.copy()

# Remove the target column from the data
y = data_copy.pop('median_house_value')

# Apply the transformations to the data
X = preprocessor.fit_transform(data_copy)

# Histograms for each numerical variable
data[numerical_cols].hist(bins=30, figsize=(20,15))
```

```
plt.show()

# Pairplot for a subset of variables
subset_of_cols = ['housing_median_age', 'total_rooms',
'median_income', 'median_house_value']
sns.pairplot(data[subset_of_cols])
plt.show()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test data
predictions = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, predictions)
mae = mean_absolute_error(y_test, predictions)
print("Linear Regression Mean Squared Error:", mse)
print("Linear Regression Mean Absolute Error:", mae)
```

C:\Users\frantzis\anaconda3\python.exe
C:\Users\frantzis\PycharmProjects\pythonProject52\main.py

Linear Regression Mean Squared Error: 4908476721.156613

Linear Regression Mean Absolute Error: 50670.73824097189