Άσκηση 6: Στην αρχική υλοποίηση, το μοντέλο εκπαιδεύτηκε με το 80% των δεδομένων και αξιολογήθηκε με το υπόλοιπο 20%. Τροποποιήστε τον κώδικα ώστε να αντιστραφούν τα σύνολα: να γίνει training με το 20% των δεδομένων και testing με το 80%.Υπολογίστε το νέο RMSE και συγκρίνετέ το με το αρχικό αποτέλεσμα. Τι παρατηρείτε;

```
# ------------------------------------------------------------
# Import required libraries
# ------------------------------------------------------------

import numpy as np                          # For numerical operations
import pandas as pd                         # For data manipulation and DataFrame operatior
import matplotlib.pyplot as plt             # For visualization
from sklearn.datasets import fetch_california_housing  # Built-in regression dataset
from sklearn.model_selection import train_test_split   # For splitting data
from sklearn.metrics import mean_squared_error         # For evaluating model performance
import xgboost as xgb                       # XGBoost library

# Set default plot size
plt.rcParams['figure.figsize'] = (7, 7)

# ------------------------------------------------------------
#  Load the California Housing dataset
# ------------------------------------------------------------

# Load the dataset as a DataFrame using scikit-learn
housing = fetch_california_housing(as_frame=True)
df = housing.frame

# Separate features (X) and target (y)
X = df.drop(columns=["MedHouseVal"])   # All predictors
y = df["MedHouseVal"]                  # Target: Median House Value

# Preview the dataset
print(" Preview of dataset:")
display(df.head())

# ------------------------------------------------------------
#  Split the data into training and testing sets
# ------------------------------------------------------------

# 20% training, 80% testing to evaluate generalization
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.8, random_state=42
)

# ------------------------------------------------------------
#  Train an XGBoost Regressor using tree-based learners
```

```
# ----------------------------------------------------------------

# Instantiate the XGBRegressor with common parameters
xg_reg = xgb.XGBRegressor(
    objective='reg:squarederror',  # Regression task
    n_estimators=100,              # Number of boosting rounds
    seed=42                        # Reproducibility
)

# Train the model on the training data
xg_reg.fit(X_train, y_train)

# Make predictions on the test set
preds = xg_reg.predict(X_test)

# Compute and print the Root Mean Squared Error
rmse = np.sqrt(mean_squared_error(y_test, preds))
print(f" Test RMSE: {rmse:.4f}")

# ----------------------------------------------------------------
#  Visualize individual trees learned by XGBoost
# ----------------------------------------------------------------

# Plot the first tree in the ensemble
fig, ax = plt.subplots(figsize=(15, 15))
xgb.plot_tree(xg_reg, num_trees=0, ax=ax)
plt.title(" Tree 0 (First Tree)")
plt.show()

# Plot the fifth tree
fig, ax = plt.subplots(figsize=(15, 15))
xgb.plot_tree(xg_reg, num_trees=4, ax=ax)
plt.title(" Tree 4")
plt.show()

# Plot the last (100th) tree in horizontal layout
fig, ax = plt.subplots(figsize=(15, 15))
xgb.plot_tree(xg_reg, num_trees=99, rankdir="LR", ax=ax)
plt.title(" Tree 99 (Last Tree, Horizontal Layout)")
plt.show()

# ----------------------------------------------------------------
#  Visualize feature importances from the trained model
# ----------------------------------------------------------------

# Plot importance scores based on feature usage frequency in splits
xgb.plot_importance(xg_reg)

# Enhance layout and appearance
plt.title("Feature Importances")
plt.tight_layout()
```

```
plt.show()
```

Άσκηση 7: Αντικαταστήστε τον XGBoost με το μοντέλο Random Forest Regressor της scikit-learn. Χρησιμοποιήστε τα ίδια training/test sets (80/20). Υπολογίστε RMSE και συγκρίνετέ το με το αντίστοιχο του XGBoost. Ποιο μοντέλο φαίνεται καλύτερο στο συγκεκριμένο dataset;

Άσκηση 7: Αντικαταστήστε τον XGBoost με το μοντέλο Random Forest Regressor της scikit-learn. Χρησιμοποιήστε τα ίδια training/test sets (80/20). Υπολογίστε RMSE και συγκρίνετέ το με το αντίστοιχο του XGBoost. Ποιο μοντέλο φαίνεται καλύτερο στο συγκεκριμένο dataset;

```
# ------------------------------------------------------------
# Import required libraries
# ------------------------------------------------------------

import numpy as np                          # For numerical operations
import pandas as pd                         # For data manipulation and DataFrame operatior
import matplotlib.pyplot as plt             # For visualization
from sklearn.datasets import fetch_california_housing  # Built-in regression dataset
from sklearn.model_selection import train_test_split   # For splitting data
from sklearn.metrics import mean_squared_error         # For evaluating model performance
import xgboost as xgb                        # XGBoost library
```

```
# Set default plot size
plt.rcParams['figure.figsize'] = (7, 7)

# ----------------------------------------------------------------
#  Load the California Housing dataset
# ----------------------------------------------------------------

# Load the dataset as a DataFrame using scikit-learn
housing = fetch_california_housing(as_frame=True)
df = housing.frame

# Separate features (X) and target (y)
X = df.drop(columns=["MedHouseVal"])   # All predictors
y = df["MedHouseVal"]                  # Target: Median House Value

# Preview the dataset
print(" Preview of dataset:")
display(df.head())

# ----------------------------------------------------------------
#  Split the data into training and testing sets
# ----------------------------------------------------------------

# 80% training, 20% testing to evaluate generalization
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# ----------------------------------------------------------------
#  Train an XGBoost Regressor using tree-based learners
# ----------------------------------------------------------------

# Instantiate the XGBRegressor with common parameters
xg_reg = xgb.XGBRegressor(
    objective='reg:squarederror',  # Regression task
    n_estimators=100,              # Number of boosting rounds
    seed=42                        # Reproducibility
)

# Train the model on the training data
xg_reg.fit(X_train, y_train)

# Make predictions on the test set
preds = xg_reg.predict(X_test)

# Compute and print the Root Mean Squared Error
rmse = np.sqrt(mean_squared_error(y_test, preds))
print(f" Test RMSE: {rmse:.4f}")

from sklearn.ensemble import RandomForestRegressor
```

```
# ----------------------------------------------------------------
#  Train a Random Forest Regressor
# ----------------------------------------------------------------

# Instantiate the model
rf_reg = RandomForestRegressor(
    n_estimators=100,
    random_state=42,
    n_jobs=-1
)

# Fit to training data
rf_reg.fit(X_train, y_train)

# Predict on test set
rf_preds = rf_reg.predict(X_test)

# Compute RMSE for Random Forest
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_preds))
print(f"Random Forest Test RMSE: {rf_rmse:.4f}")


# # ----------------------------------------------------------------
# #  Visualize individual trees learned by XGBoost
# # ----------------------------------------------------------------

# # Plot the first tree in the ensemble
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=0, ax=ax)
# plt.title(" Tree 0 (First Tree)")
# plt.show()

# # Plot the fifth tree
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=4, ax=ax)
# plt.title(" Tree 4")
# plt.show()

# # Plot the last (100th) tree in horizontal layout
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=99, rankdir="LR", ax=ax)
# plt.title(" Tree 99 (Last Tree, Horizontal Layout)")
# plt.show()

# ----------------------------------------------------------------
#  Visualize feature importances from the trained model
# ----------------------------------------------------------------

# # Plot importance scores based on feature usage frequency in splits
# xgb.plot_importance(xg_reg)
```

```
# # Enhance layout and appearance
# plt.title("Feature Importances")
# plt.tight_layout()
# plt.show()


# ----------------------------------------------------------------
#  Συγκριτική αξιολόγηση
# ----------------------------------------------------------------

print("\nComparison:")
print(f"XGBoost RMSE       : {rmse:.4f}")
print(f"Random Forest RMSE : {rf_rmse:.4f}")

if rf_rmse < rmse:
    print("Random Forest είχε RMSE is lower.")
elif rf_rmse > rmse:
    print("XGBoost RMSE is lower.")
else:
    print("Both models have the same RMSE.")
```

Άσκηση 8: Αφαιρέστε από το πλήρες dataset το χαρακτηριστικό MedInc, το οποίο εμφανίζεται ως το πιο σημαντικό στην απεικόνιση feature importance. Εκπαιδεύστε νέο μοντέλο με όλα τα features εκτός του MedInc. Υπολογίστε το RMSE. Συγκρίνετέ το με την πλήρη έκδοση του dataset.

```
# ----------------------------------------------------------------
# Import required libraries
# ----------------------------------------------------------------
```

```
# ------------------------------------------------------------

import numpy as np                            # For numerical operations
import pandas as pd                           # For data manipulation and DataFrame operatior
import matplotlib.pyplot as plt               # For visualization
from sklearn.datasets import fetch_california_housing  # Built-in regression dataset
from sklearn.model_selection import train_test_split   # For splitting data
from sklearn.metrics import mean_squared_error         # For evaluating model performance
import xgboost as xgb                          # XGBoost library

# Set default plot size
plt.rcParams['figure.figsize'] = (7, 7)

# ------------------------------------------------------------
#  Load the California Housing dataset
# ------------------------------------------------------------

# Load the dataset as a DataFrame using scikit-learn
housing = fetch_california_housing(as_frame=True)
df = housing.frame

# Separate features (X) and target (y)
X = df.drop(columns=["MedHouseVal"])   # All predictors
y = df["MedHouseVal"]                  # Target: Median House Value

# Preview the dataset
print(" Preview of dataset:")
display(df.head())

# Νέο X χωρίς MedInc
X_no_medinc = X.drop(columns=["MedInc"])

# ------------------------------------------------------------
#  Split the data into training and testing sets
# ------------------------------------------------------------

# 80% training, 20% testing to evaluate generalization
X_train_nomed, X_test_nomed, y_train_nomed, y_test_nomed = train_test_split(
    X_no_medinc, y, test_size=0.2, random_state=42
)

# ------------------------------------------------------------
#  Train an XGBoost Regressor using tree-based learners
# ------------------------------------------------------------

# Instantiate the XGBRegressor with common parameters
xg_reg_nomed = xgb.XGBRegressor(
    objective='reg:squarederror',  # Regression task
    n_estimators=100,              # Number of boosting rounds
    seed=42                        # Reproducibility
)
```

```
# Train the model on the training data
xg_reg_nomed.fit(X_train_nomed, y_train_nomed)

# Make predictions on the test set
preds_nomed = xg_reg_nomed.predict(X_test_nomed)

# Compute and print the Root Mean Squared Error
rmse_nomed = np.sqrt(mean_squared_error(y_test_nomed, preds_nomed))
print(f" Test RMSE: {rmse_nomed:.4f}")

# # ----------------------------------------------------------------
# #  Visualize individual trees learned by XGBoost
# # ----------------------------------------------------------------

# # Plot the first tree in the ensemble
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=0, ax=ax)
# plt.title(" Tree 0 (First Tree)")
# plt.show()

# # Plot the fifth tree
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=4, ax=ax)
# plt.title(" Tree 4")
# plt.show()

# # Plot the last (100th) tree in horizontal layout
# fig, ax = plt.subplots(figsize=(15, 15))
# xgb.plot_tree(xg_reg, num_trees=99, rankdir="LR", ax=ax)
# plt.title(" Tree 99 (Last Tree, Horizontal Layout)")
# plt.show()

# ----------------------------------------------------------------
#  Visualize feature importances from the trained model
# ----------------------------------------------------------------

# # Plot importance scores based on feature usage frequency in splits
# xgb.plot_importance(xg_reg)

# # Enhance layout and appearance
# plt.title("Feature Importances")
# plt.tight_layout()
# plt.show()

# ----------------------------------------------------------------
#  Σύγκριση RMSE με πλήρες dataset
# ----------------------------------------------------------------

print("\nΣύγκριση επιδόσεων:")
print(f"Με όλα τα features: {rmse: 4f}")
```

```
print(f"Με όλα τα features: {rmse:.4f}")
print(f"Χωρίς MedInc: {rmse_nomed:.4f}")

if rmse_nomed > rmse:
    print("Η αφαίρεση του MedInc έφερε υψηλότερο RMSE.")
elif rmse_nomed < rmse:
    print("Η αφαίρεση του MedInc έφερε χαμηλότερο RMSE.")
else:
    print("Καμία διαφορά στην απόδοση μετά την αφαίρεση του MedInc.")
```