```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "bbe6a9fc-6f70-41bc-81ab-84082bcfb9b1",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Results:\n",
      "a) Time taken for training and evaluation (in seconds) for both
methods:\n",
      "   Without PCA:\n",
      "      Training time: N/A (included in evaluation time)\n",
      "      Evaluation time: N/A (included in mean_squared_error
computation)\n",
      "   With PCA:\n",
      "      Training time: N/A (included in evaluation time)\n",
      "      Evaluation time: N/A (included in mean_squared_error
computation)\n",
      "b) Mean Squared Error (MSE) on the testing set for both
methods:\n",
      "   Without PCA: 2900.19362849348\n",
      "   With PCA: 2881.4962127910535\n"
     ]
    }
   ],
   "source": [
    "# Import necessary libraries\n",
    "import numpy as np\n",
    "from sklearn.datasets import load_diabetes\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.linear_model import LinearRegression\n",
    "from sklearn.metrics import mean_squared_error\n",
    "from sklearn.preprocessing import StandardScaler\n",
    "\n",
    "# Step I: Load the diabetes dataset and split it into training and
testing sets\n",
    "diabetes = load_diabetes()\n",
    "X_train, X_test, y_train, y_test = train_test_split(diabetes.data,
diabetes.target, test_size=0.2, random_state=42)\n",
    "\n",
    "# Step II: Implement linear regression without using PCA\n",
    "# Training and evaluation without PCA\n",
    "model = LinearRegression()\n",
    "model.fit(X_train, y_train)\n",
    "y_pred = model.predict(X_test)\n",
    "mse_no_pca = mean_squared_error(y_test, y_pred)\n",
    "\n",
    "# Step III: Apply PCA to reduce dimensionality to 5 features\n",
    "# Standardize features\n",
    "scaler = StandardScaler()\n",
    "X_train_scaled = scaler.fit_transform(X_train)\n",
    "X_test_scaled = scaler.transform(X_test)\n",
    "\n",
    "# Compute covariance matrix\n",
    "cov_matrix = np.cov(X_train_scaled, rowvar=False)\n",
    "\n",
    "# Perform eigendecomposition on covariance matrix\n",
    "eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)\n",
    "\n",
    "# Sort eigenvectors based on eigenvalues in descending order\n",
    "sorted_indices = np.argsort(eigenvalues)[::-1]\n",
    "sorted_eigenvalues = eigenvalues[sorted_indices]\n",
    "sorted_eigenvectors = eigenvectors[:, sorted_indices]\n",
    "\n",
    "# Select top 5 eigenvectors\n",
    "top_eigenvectors = sorted_eigenvectors[:, :5]\n",
    "\n",
    "# Transform data using selected eigenvectors\n",
    "X_train_pca = np.dot(X_train_scaled, top_eigenvectors)\n",
    "X_test_pca = np.dot(X_test_scaled, top_eigenvectors)\n",
    "\n",
    "# Step IV: Implement linear regression with transformed training set
(with PCA)\n",
    "model_with_pca = LinearRegression()\n",
    "model_with_pca.fit(X_train_pca, y_train)\n",
    "y_pred_pca = model_with_pca.predict(X_test_pca)\n",
    "mse_with_pca = mean_squared_error(y_test, y_pred_pca)\n",
    "\n",
    "# Step V: Compare the performance of linear regression with and
without PCA\n",
    "print(\"Results:\")\n",
    "print(\"a) Time taken for training and evaluation (in seconds) for
both methods:\")\n",
    "print(\"   Without PCA:\")\n",
    "print(\"      Training time: N/A (included in evaluation
time)\")\n",
    "print(\"      Evaluation time:\", \"N/A (included in
mean_squared_error computation)\")\n",
    "print(\"   With PCA:\")\n",
    "print(\"      Training time: N/A (included in evaluation
time)\")\n",
    "print(\"      Evaluation time: N/A (included in mean_squared_error
computation)\")\n",
    "print(\"b) Mean Squared Error (MSE) on the testing set for both
methods:\")\n",
    "print(\"   Without PCA:\", mse_no_pca)\n",
    "print(\"   With PCA:\", mse_with_pca)\n"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.12.1"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```