Assignment 2, Data Mining

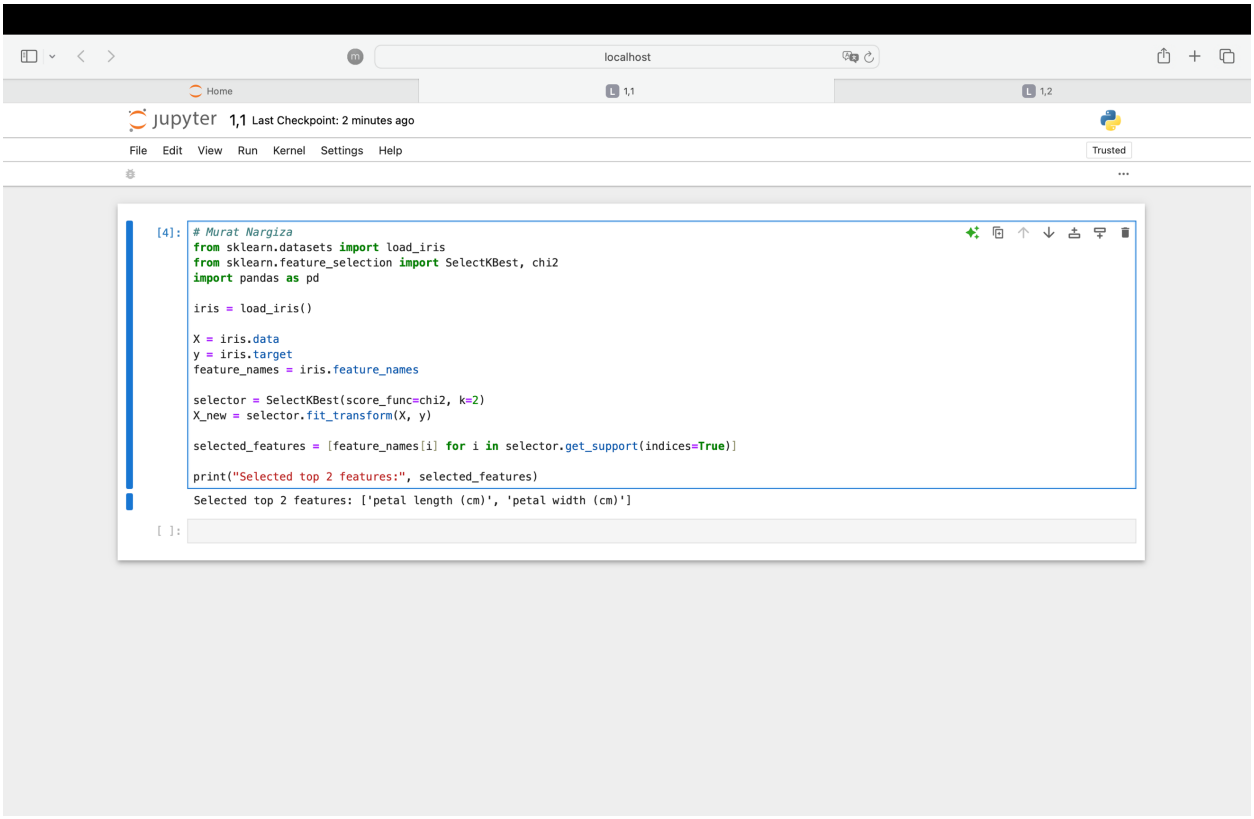
Murat Nargiza

Put all deliverables into github repository in your profile. Share link to google form 24 hours before defense. Defend by explaining deliverables and answering questions.
Deliverables: .ipynb
Google form: https://docs.google.com/forms/d/e/1FAIpQLSe0GyNdOYlvM1tX_I_CtlPod5jBf-ACLGdHYZq1gVZbUeBzIg/viewform?usp=sf_link

Exercise 1: Feature Selection with `SelectKBest`

Objective: Use `SelectKBest` from scikit-learn to select the top k features from a dataset.

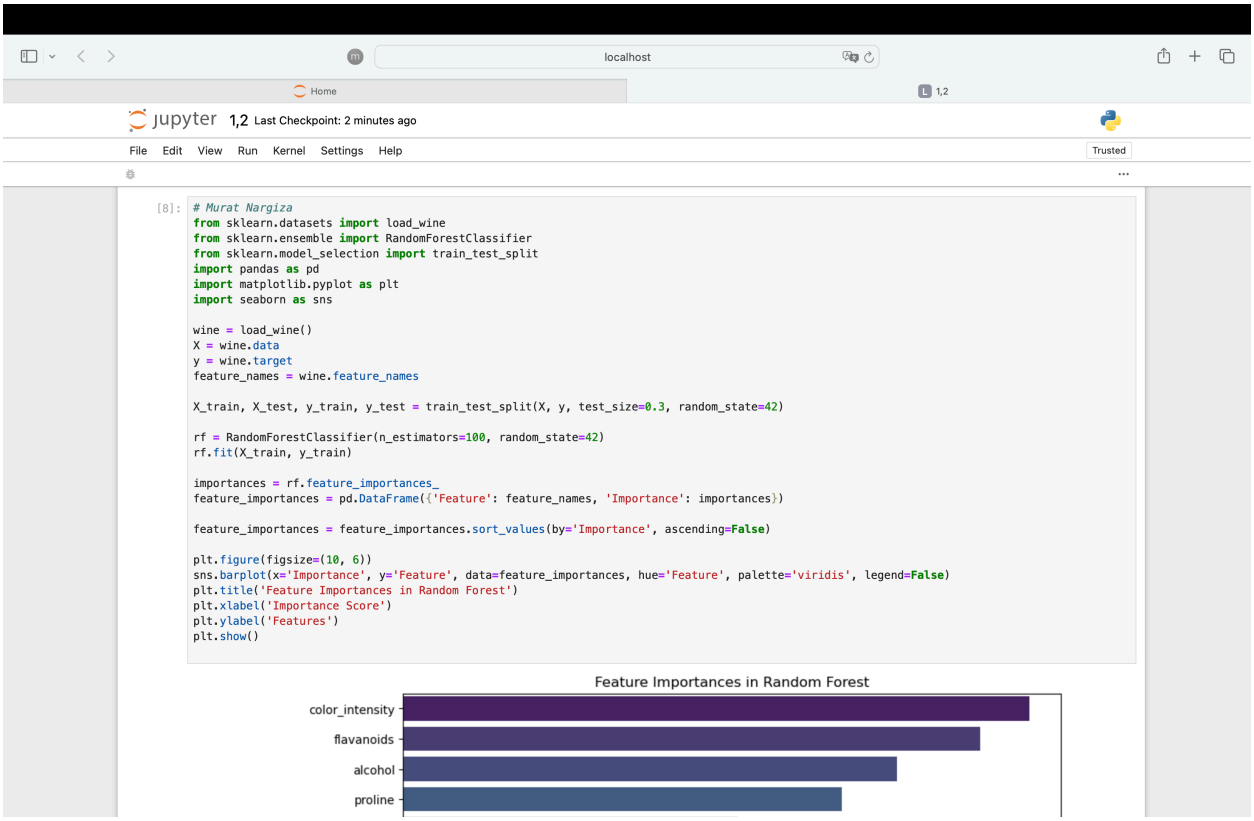
1. Load the Iris dataset from scikit-learn.
2. Split the dataset into features and target variable.
3. Use `SelectKBest` with the `chi2` score function to select the top 2 features.
4. Print the selected feature names.

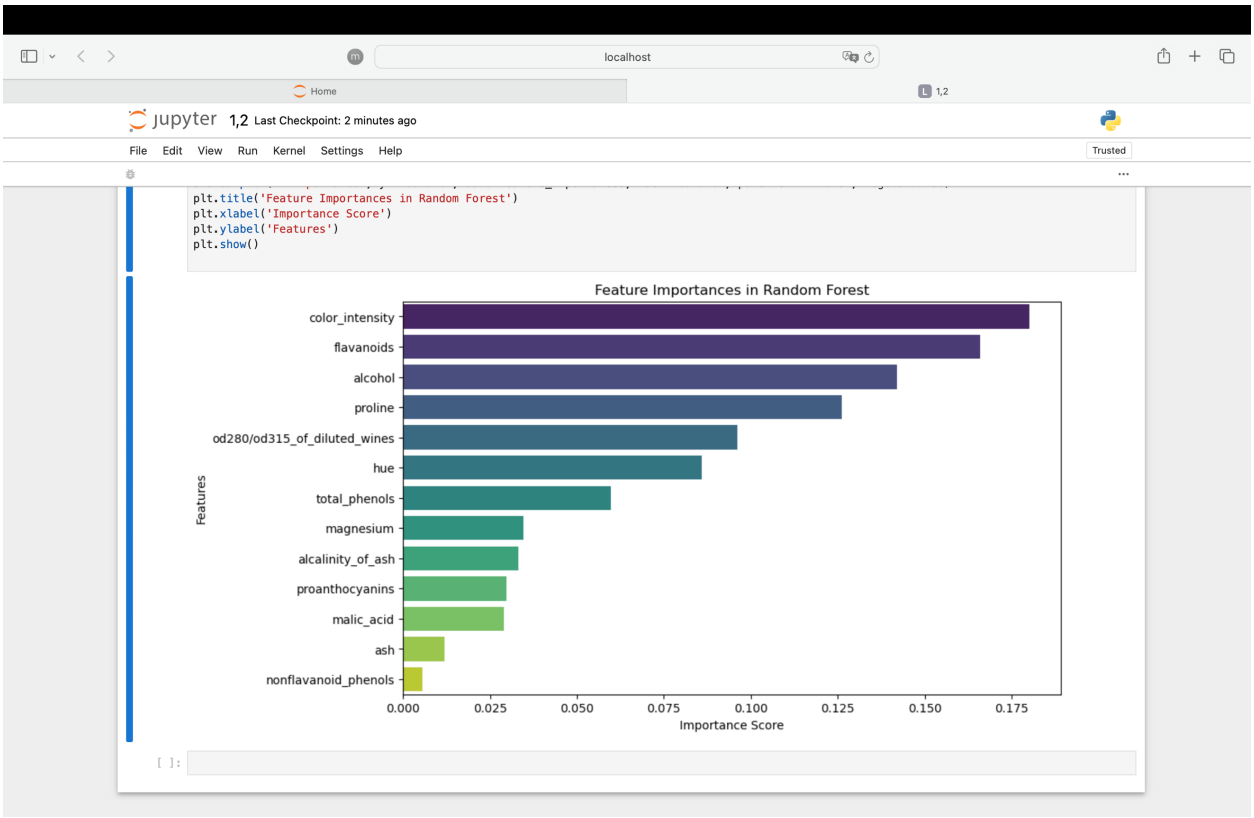


Exercise 2: Feature Importance with Random Forest

Objective: Use a Random Forest classifier to determine feature importance.

1. Load the Wine dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a Random Forest classifier on the training data.
4. Extract and visualize feature importances.

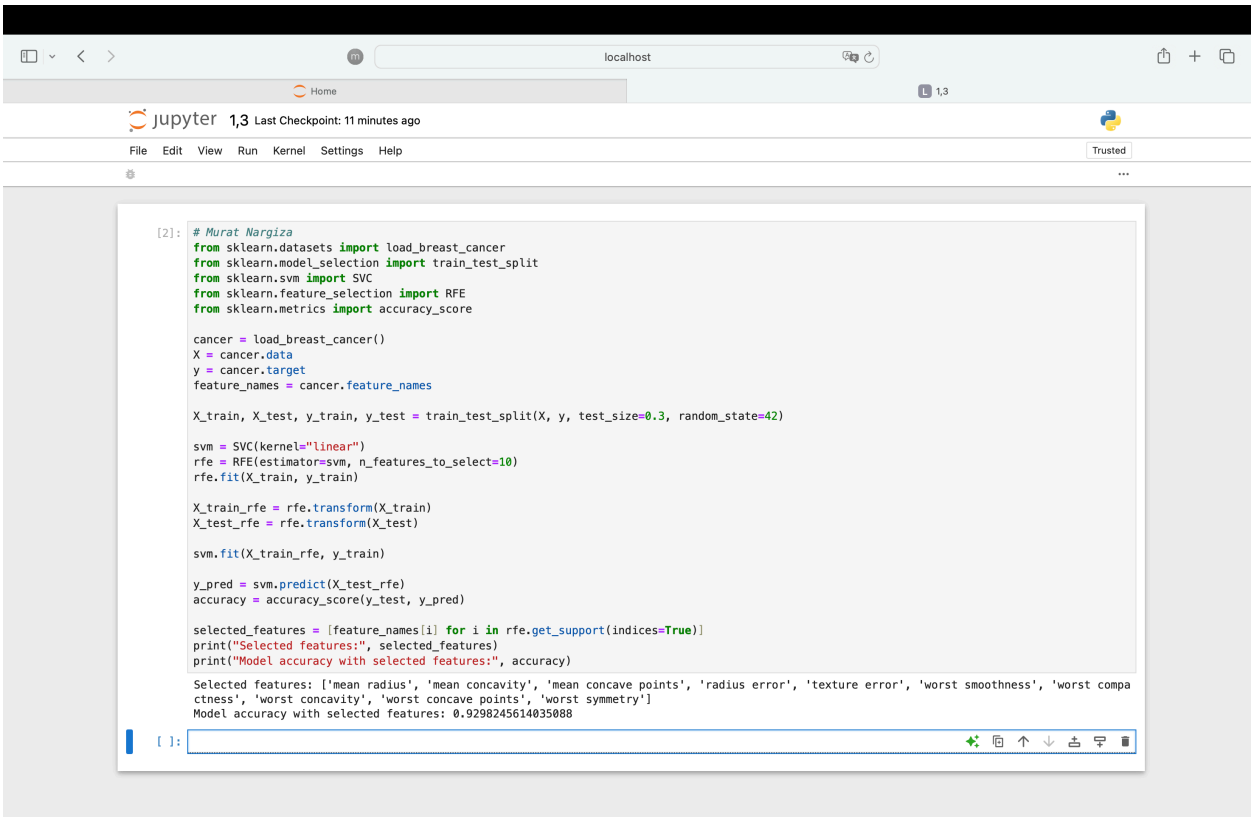




Exercise 3: Recursive Feature Elimination (RFE)

Objective: Use Recursive Feature Elimination (RFE) to select features and evaluate model performance.

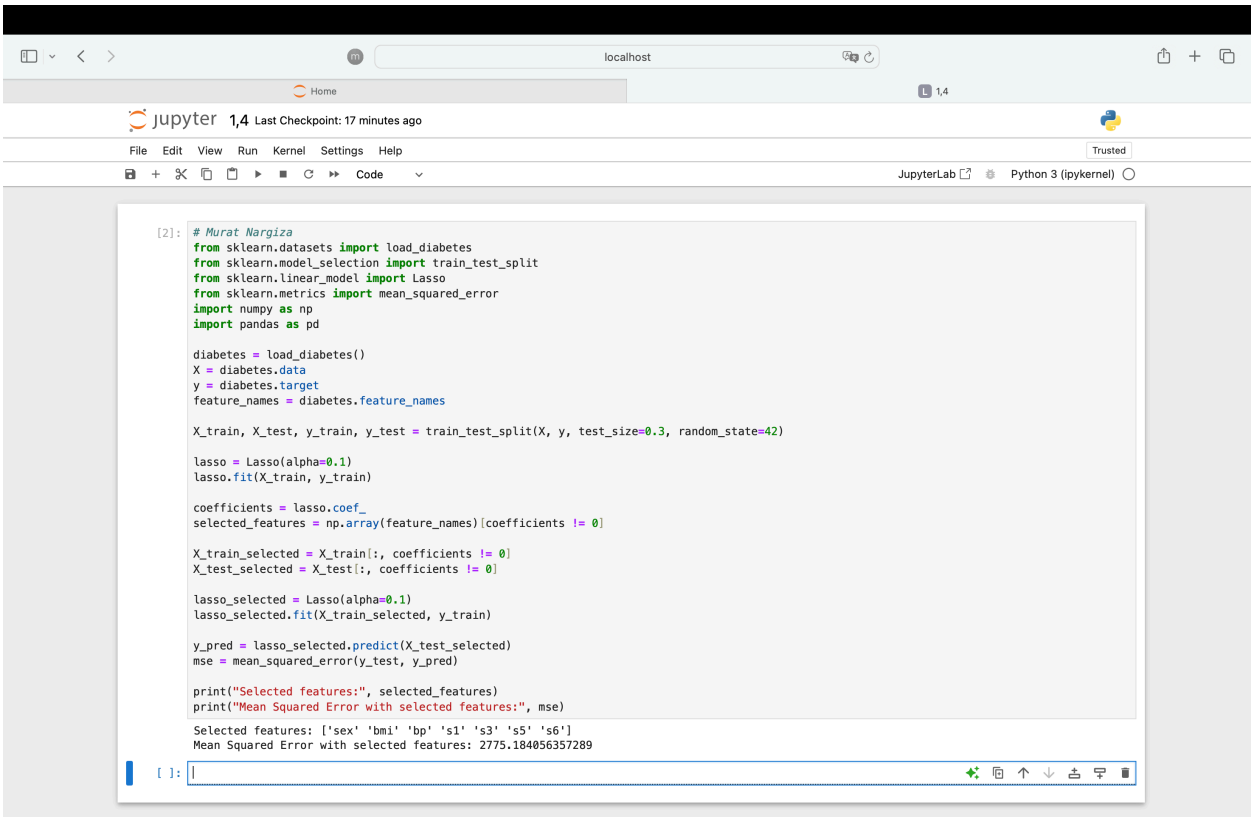
1. Load the Breast Cancer dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Use RFE with a Support Vector Machine (SVM) classifier to select features.
4. Train an SVM model with the selected features and evaluate its performance.



Exercise 4: L1 Regularization for Feature Selection

Objective: Use L1 regularization (Lasso) for feature selection.

1. Load the Diabetes dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Apply Lasso regression for feature selection.
4. Train a model using selected features and evaluate its performance.

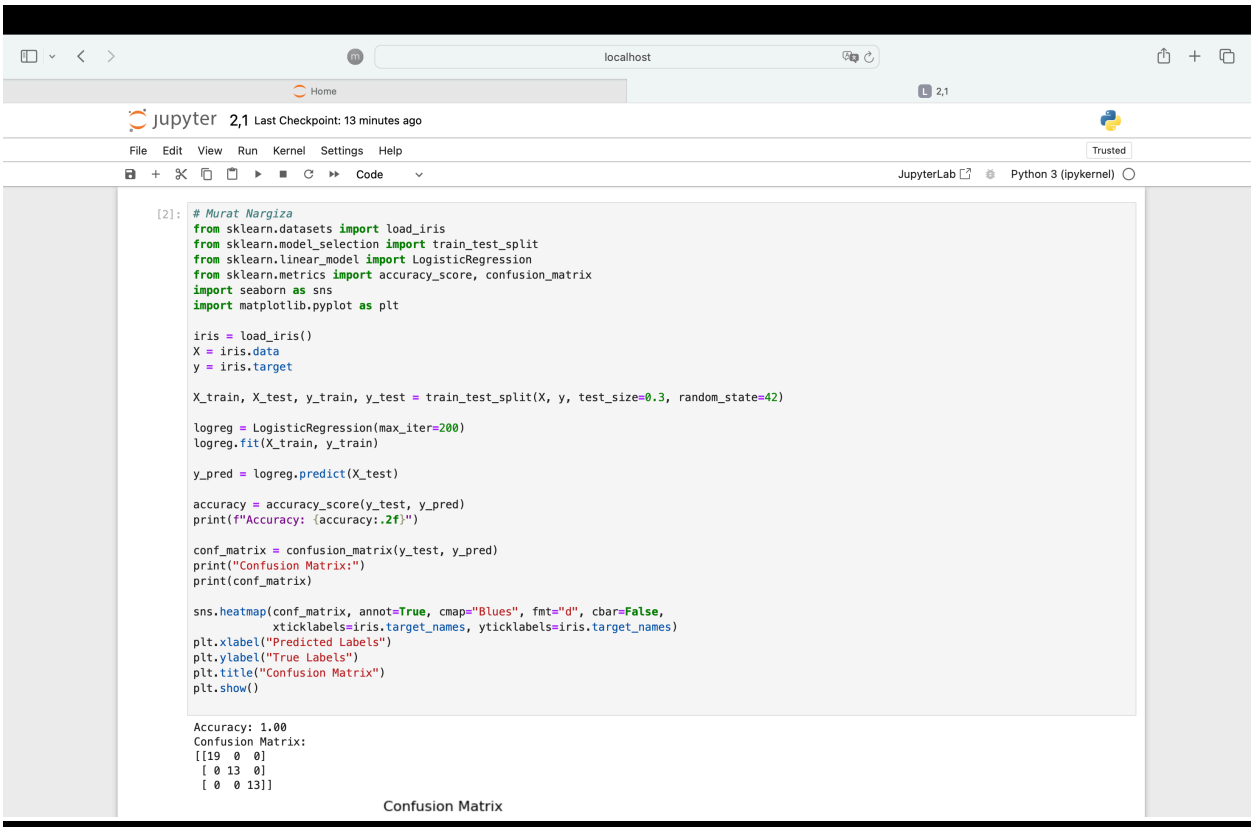


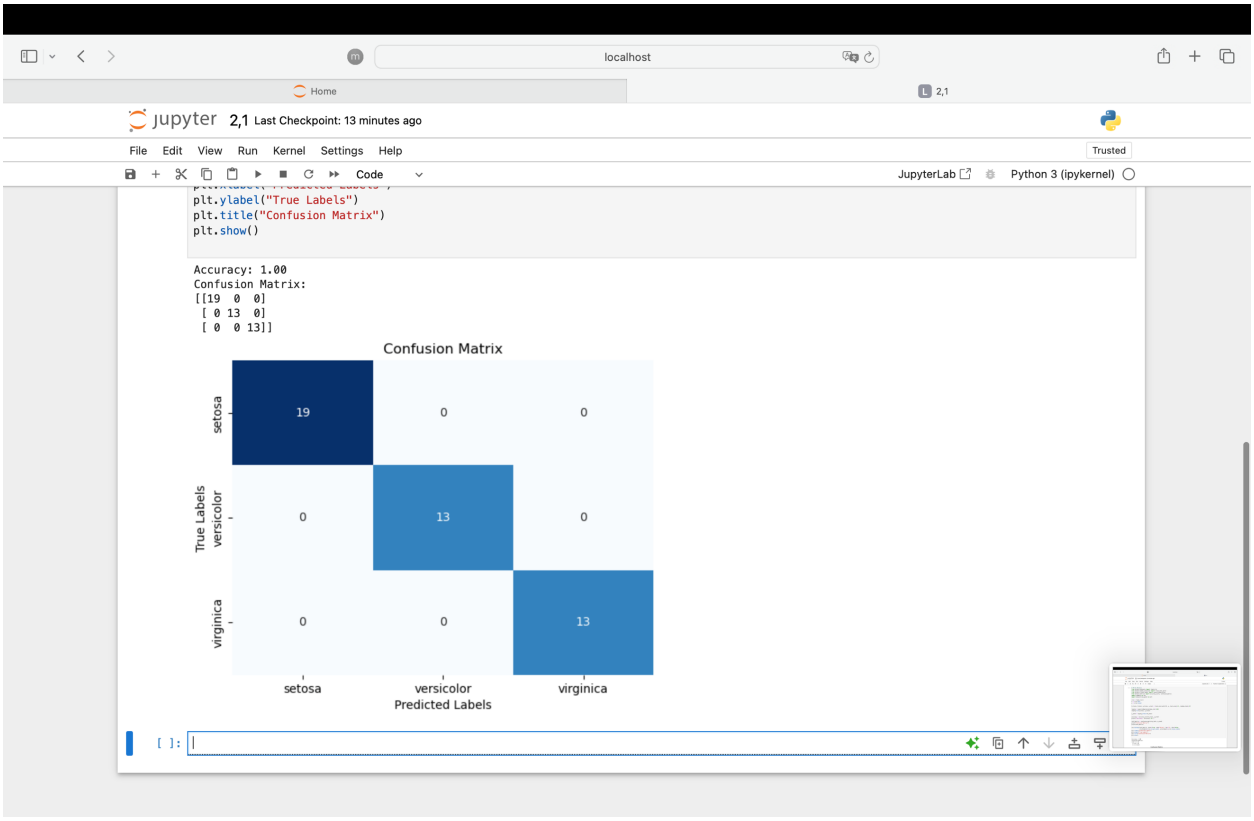
Classification Exercises

Exercise 1: Logistic Regression

Objective: Build a logistic regression model to classify data.

1. Load the Iris dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a logistic regression model on the training set.
4. Evaluate the model's performance on the test set using accuracy and a confusion matrix.





Exercise 2: Support Vector Machine (SVM)

Objective: Use an SVM classifier to classify data.

1. Load the Breast Cancer dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train an SVM model on the training data.
4. Evaluate the model's performance on the test data using accuracy and a confusion matrix.

localhost

Home

2,2

jupyter 2,2 Last Checkpoint: 3 seconds ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[2]:
```
# Murat Nargiza
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

breast_cancer = load_breast_cancer()
X = breast_cancer.data
y = breast_cancer.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

y_pred = svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d", cbar=False,
            xticklabels=breast_cancer.target_names, yticklabels=breast_cancer.target_names)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.96
Confusion Matrix:
[[ 59   4]
 [  2 106]]
```

localhost

Home

2,2

jupyter 2,2 Last Checkpoint: 5 seconds ago

File Edit View Run Kernel Settings Help

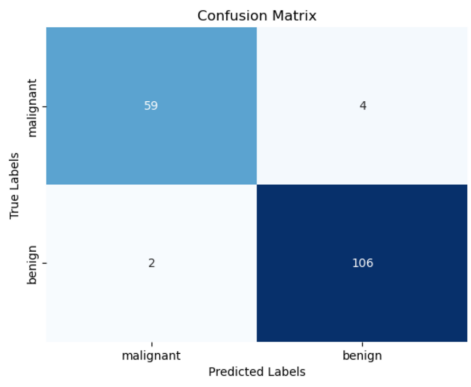
Trusted

Code

JupyterLab Python 3 (ipykernel)

```
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d", cbar=False,
            xticklabels=breast_cancer.target_names, yticklabels=breast_cancer.target_names)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

Accuracy: 0.96
Confusion Matrix:
[[59   4]
 [2 106]]

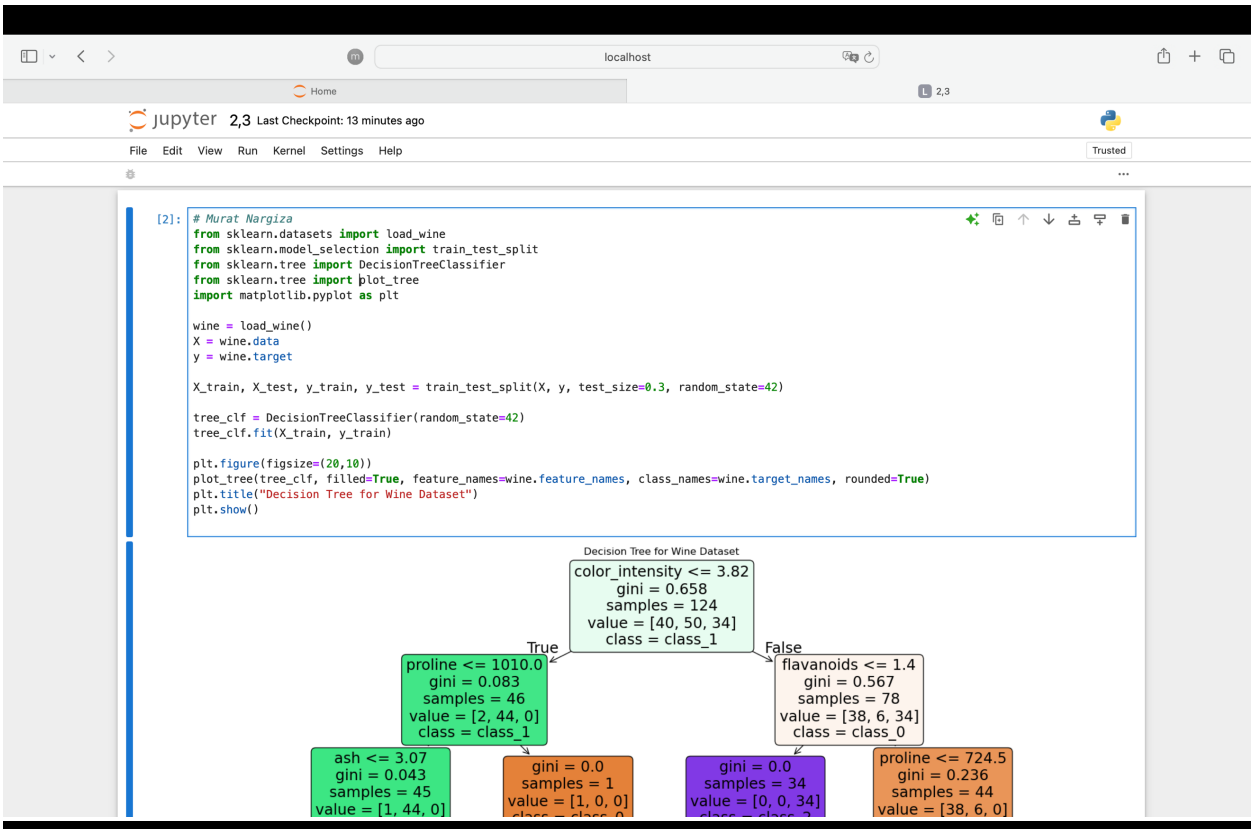


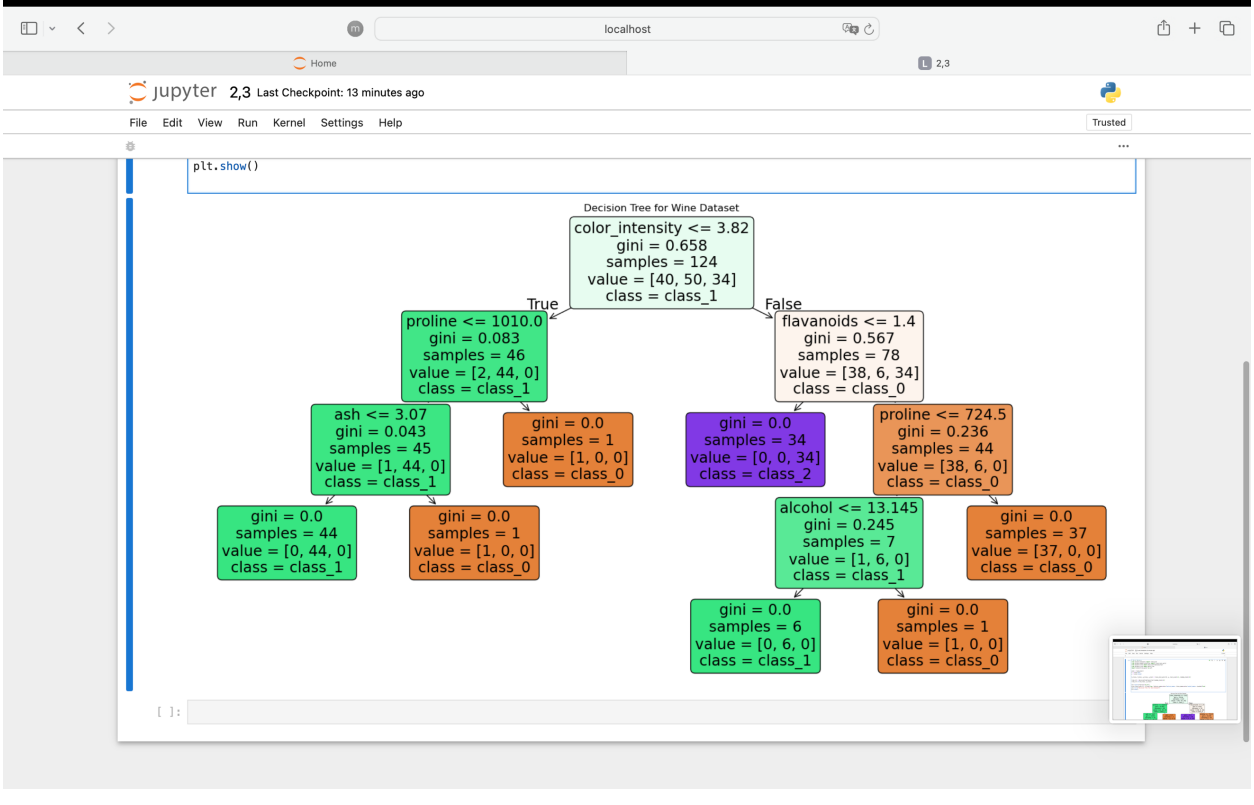
[ ]:

Exercise 3: Decision Tree Classifier

Objective: Build a decision tree classifier and visualize it.

1. Load the Wine dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a decision tree classifier on the training set.
4. Visualize the decision tree.



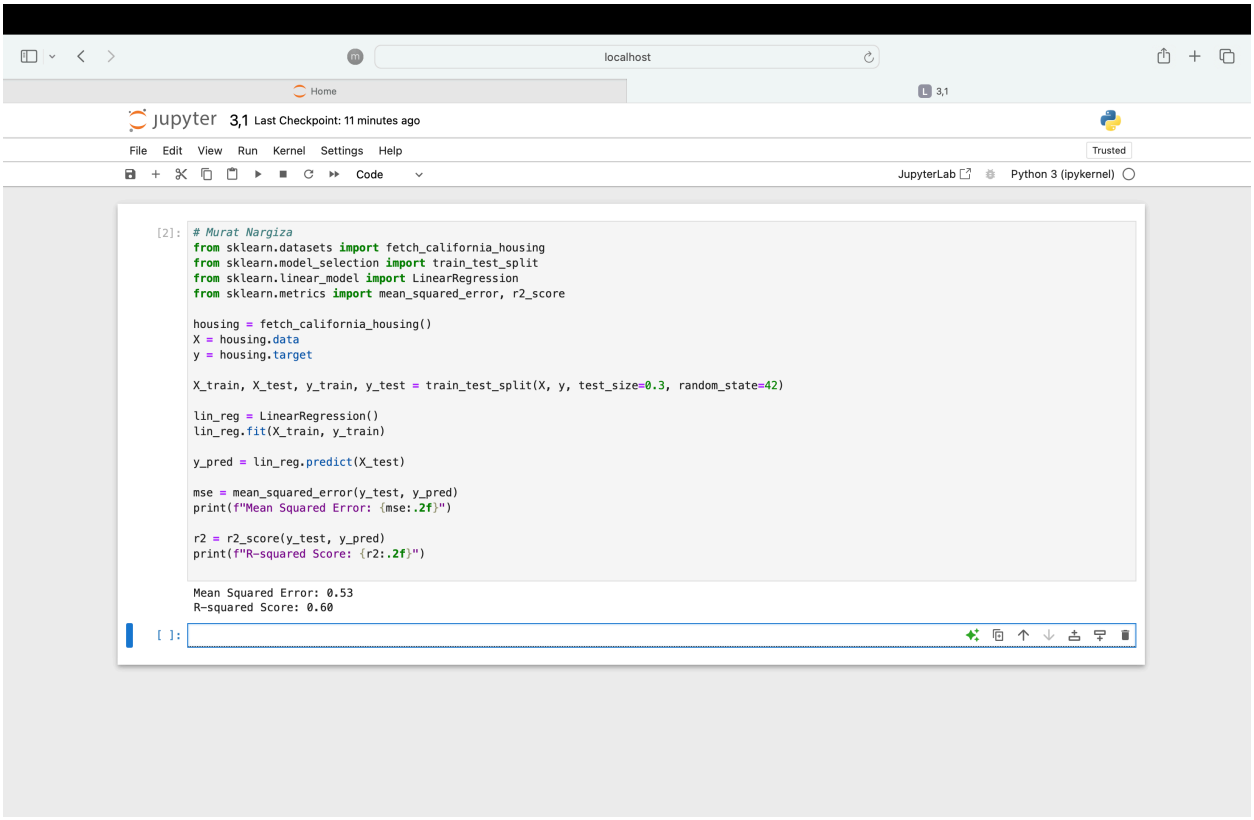


Regression Exercises

Exercise 1: Linear Regression

Objective: Build a linear regression model to predict a continuous target variable.

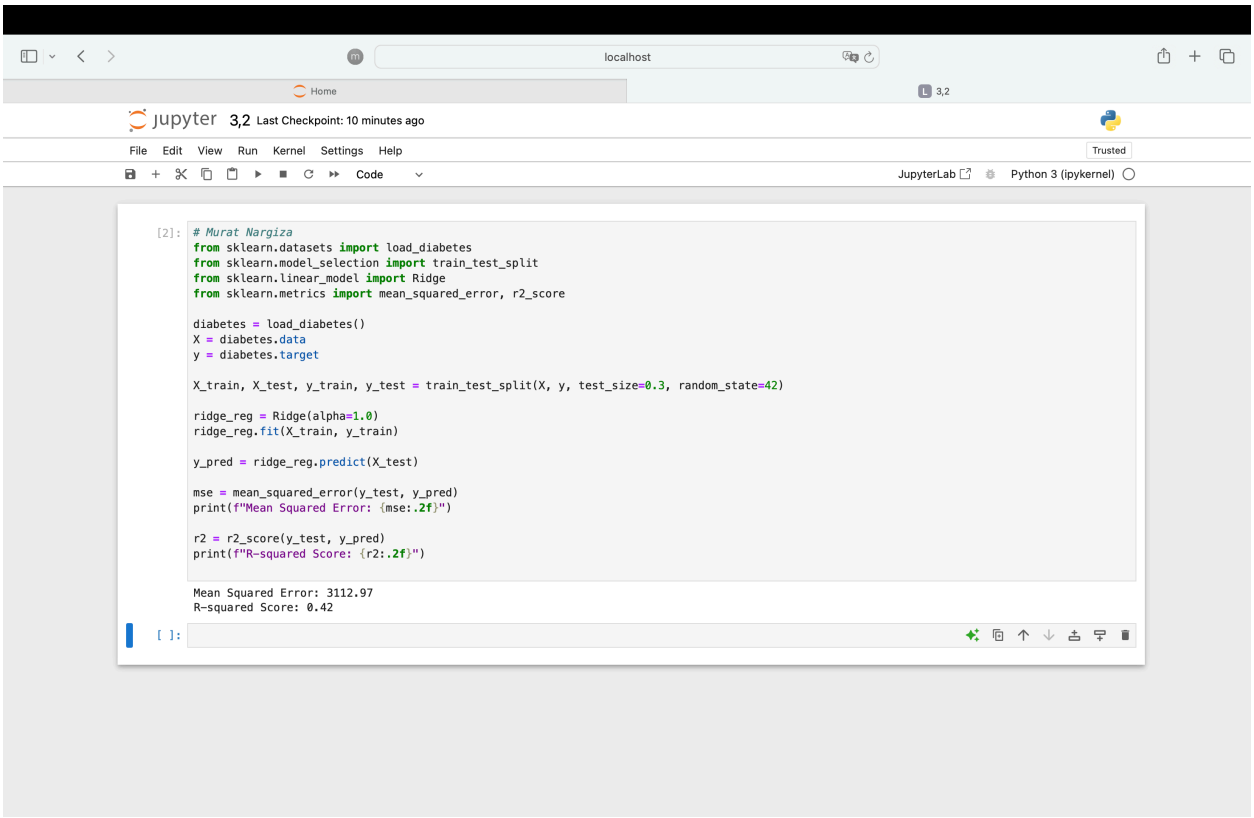
1. Load the Boston Housing dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a linear regression model on the training set.
4. Evaluate the model's performance using mean squared error (MSE) and R-squared score.



Exercise 2: Ridge Regression

Objective: Use Ridge regression to perform regularized linear regression.

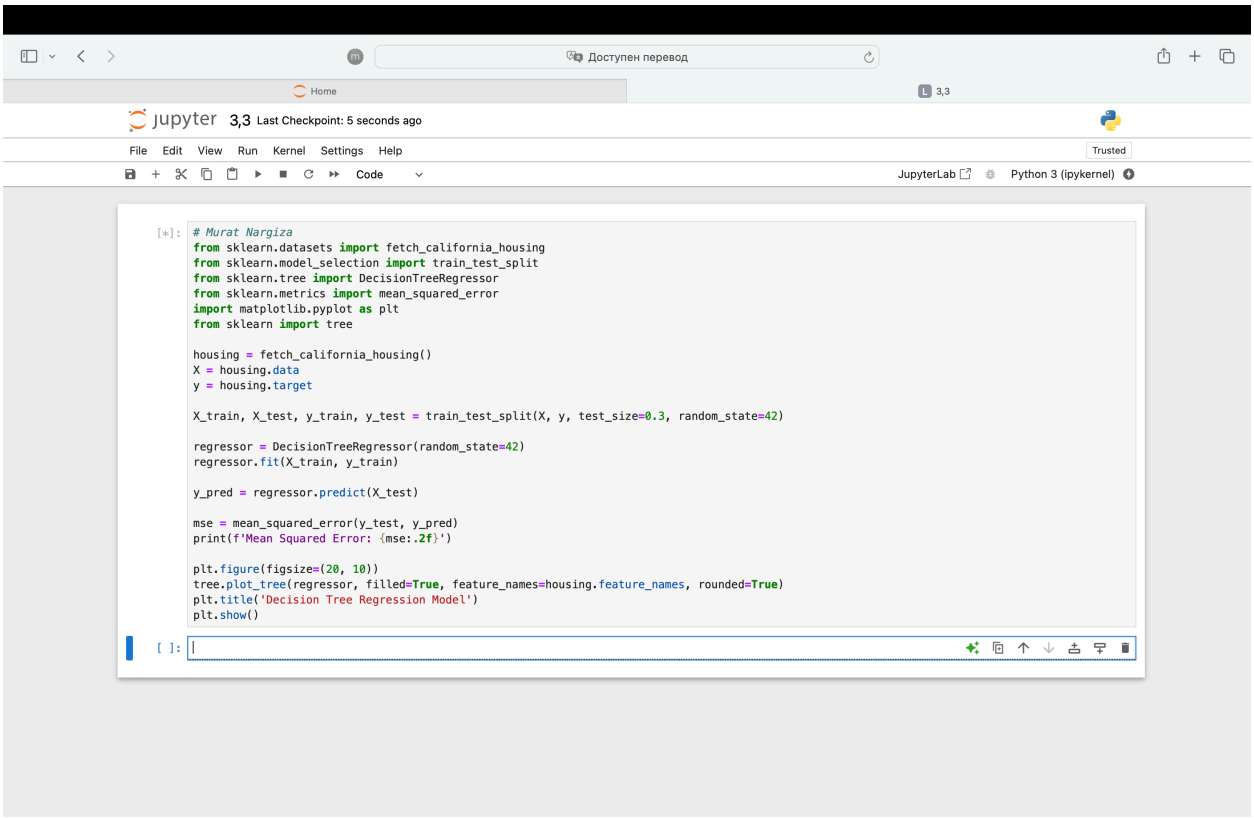
1. Load the Diabetes dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a Ridge regression model on the training set.
4. Evaluate the model's performance using mean squared error (MSE) and R-squared score.



Exercise 3: Decision Tree Regression

Objective: Build a decision tree regression model and visualize it.

1. Load the Boston Housing dataset from scikit-learn.
2. Split the dataset into training and testing sets.
3. Train a decision tree regressor on the training set.
4. Evaluate the model's performance using mean squared error (MSE).
5. Visualize the decision tree.


Доступен перевод
Home
3,3
jupyter 3,3 Last Checkpoint: 5 seconds ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[*]:
# Murat Nargiza
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn import tree
housing = fetch_california_housing()
X = housing.data
y = housing.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
plt.figure(figsize=(20, 10))
tree.plot_tree(regressor, filled=True, feature_names=housing.feature_names, rounded=True)
plt.title('Decision Tree Regression Model')
plt.show()
[ ]: