Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

Task 01

1. Load the Dataset

To import a housing dataset, use Scikit-learn's California Housing Prices dataset or a custom CSV file with features like square footage and bedrooms.

Pythontask.py > ...

```python
import pandas as pd
from sklearn.datasets import fetch_california_housing

# Example: Loading a dataset from sklearn
housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['MedHouseVal'] = housing.target
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

If you are using a CSV file:

df = pd.read_csv('path_to_your_file.csv')

2. Data Preprocessing

Handle Missing Values:

The task involves identifying and addressing missing values through imputation methods like mean, median, or mode, or by removing rows or columns.

Feature Scaling:

The task involves normalizing or standardizing features to ensure they are on a similar scale.

Pythontask.py > ...

```python
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Handle missing values
imputer = SimpleImputer(strategy='median')
df = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)

# Scale features
scaler = StandardScaler()
df[df.columns] = scaler.fit_transform(df)
```

3. Feature Selection

Choose the most significant factors that are likely to influence house prices, such as square footage, number of bedrooms, and location.

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

Pythontask.py > ...

```python
X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']
```

4. Model Selection

Select regression techniques such as Linear Regression or Decision Trees.

Pythontask.py > ...

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

# Split 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)

# Initialize models
linear_reg = LinearRegression()
tree_reg = DecisionTreeRegressor()
```

Ln 11, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.5 64-bit Codeium: {...}

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

5. Model Training

The model(s) will be trained using the training data.

```python
# Train Linear Regression model
linear_reg.fit(X_train, y_train)

# Train Decision Tree model
tree_reg.fit(X_train, y_train)
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

6. Model Evaluation

The models' performance can be compared using metrics like Mean Squared Error (MSE) on the test data.

```python
from sklearn.metrics import mean_squared_error

# Predict using both models
linear_pred = linear_reg.predict(X_test)
tree_pred = tree_reg.predict(X_test)

# Calculate MSE
linear_mse = mean_squared_error(y_test, linear_pred)
tree_mse = mean_squared_error(y_test, tree_pred)

print(f"Linear Regression MSE: {linear_mse}")
print(f"Decision Tree MSE: {tree_mse}")
```

7. Model Deployment

To deploy the model, use a framework like Flask or Django to create an API for price prediction by requiring user input of features.

8. Conclusion

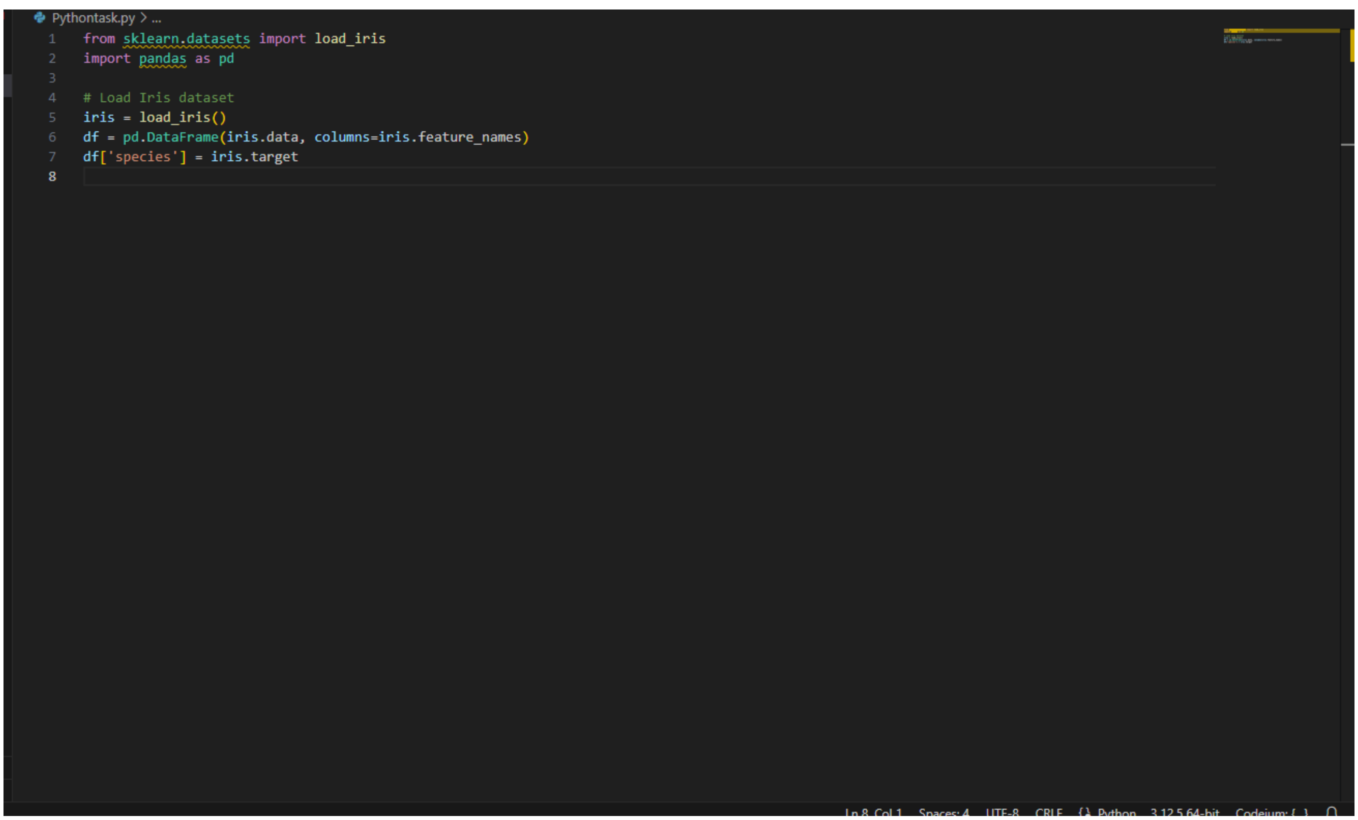
The task involves comparing the MSE of two models and determining the most effective one, with the possibility of tuning hyperparameters for improved performance.

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

Task 02

1. Load the Iris Dataset

The Iris dataset is a widely-used and easily accessible dataset in the Scikit-learn library.

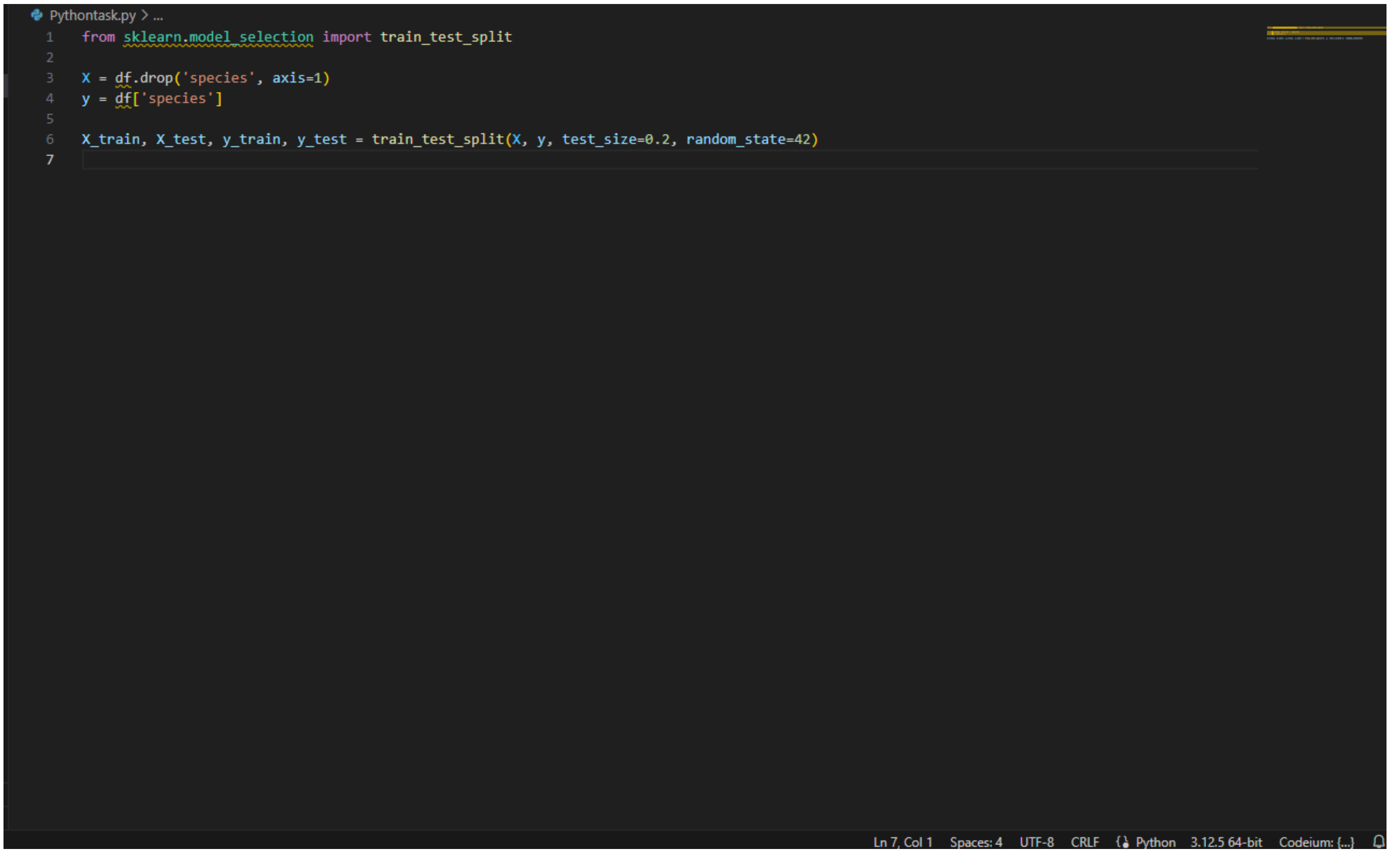


Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

2. Preprocess the Dataset

Split the Data:

The dataset should be divided into training and testing sets to assess the model's performance.




```python
from sklearn.model_selection import train_test_split

X = df.drop('species', axis=1)
y = df['species']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

3. Select a Classification Algorithm

Select a classification algorithm like Logistic Regression, Decision Trees, or Support Vector Machine (SVM) for classification.

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

# Initialize classifiers
log_reg = LogisticRegression(max_iter=200)
tree_clf = DecisionTreeClassifier()
svm_clf = SVC()
```

4. Train the Model

The selected model(s) will be trained using the training data.

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

Pythontask.py

```python
# Train Logistic Regression model
log_reg.fit(X_train, y_train)

# Train Decision Tree model
tree_clf.fit(X_train, y_train)

# Train SVM model
svm_clf.fit(X_train, y_train)

```

5. Evaluate the Model

The model's accuracy and performance on the test set will be assessed using metrics such as accuracy, precision, recall, and F1-score.

Pythontask.py > ...

```python
from sklearn.metrics import accuracy_score, classification_report

# Predict using the models
log_reg_pred = log_reg.predict(X_test)
tree_clf_pred = tree_clf.predict(X_test)
svm_clf_pred = svm_clf.predict(X_test)

# Evaluate Logistic Regression
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_reg_pred))
print("Logistic Regression Report:\n", classification_report(y_test, log_reg_pred, target_names=iris.target_names))

# Evaluate Decision Tree
print("Decision Tree Accuracy:", accuracy_score(y_test, tree_clf_pred))
print("Decision Tree Report:\n", classification_report(y_test, tree_clf_pred, target_names=iris.target_names))

# Evaluate SVM
print("SVM Accuracy:", accuracy_score(y_test, svm_clf_pred))
print("SVM Report:\n", classification_report(y_test, svm_clf_pred, target_names=iris.target_names))

```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

6. Compare Models

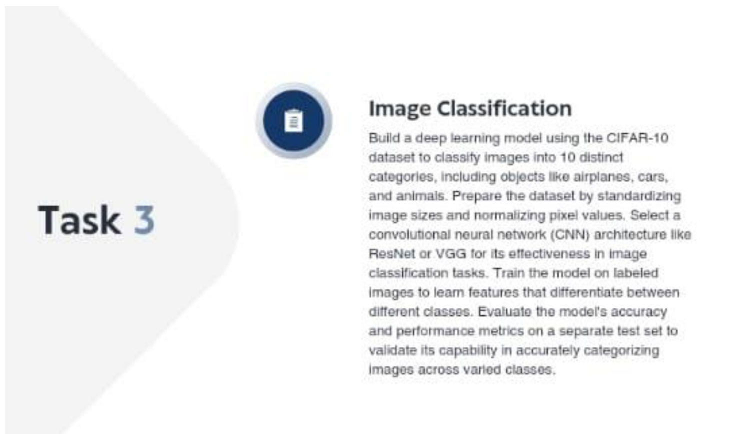
The study compares the performance of various models based on accuracy and other evaluation metrics.

7. Conclusion

The evaluation results will be used to select the most effective model for classifying Iris flowers.

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

Task 03



1. Set Up the Environment

Install Required Libraries:

pip install tensorflow keras numpy matplotlib

Import Necessary Modules:

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

2. Load and Prepare the CIFAR-10 Dataset

Pythontask.py

```python
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

3. Visualize the Dataset (Optional)

Pythontask.py > ...

```python
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

4. Build the CNN Model

Pythontask.py > ...

```python
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

5. Compile the Model

```
PythonTask.py
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

6. Train the Model

```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

7. Evaluate the Model

```python
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f'\nTest accuracy: {test_acc}')
```

8. Plot the Accuracy and Loss Over Time

```python
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

Name M Hammad Hussain
Email hammadhussain1972@gmail.com
Domain Machine learning

9. Make Predictions

```
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)

# Example prediction
print(class_names[np.argmax(predictions[0])])
```

10. Save the Model

model.save('cifar10_model.h5')