

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

Task 01

Task 1



House Price Prediction

Develop a machine learning model using a housing dataset to predict house prices based on features like square footage and number of bedrooms. Preprocess the data to handle missing values and scale numerical features. Train the model using regression techniques such as Linear Regression or Decision Trees. Evaluate its accuracy using metrics like Mean Squared Error on test data to ensure reliable predictions. This model aims to assist in estimating house prices accurately for new properties.

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.

```
Python task.py > ...
1 import pandas as pd
2 from sklearn.datasets import fetch_california_housing
3
4 # Example: Loading a dataset from sklearn
5 housing = fetch_california_housing()
6 df = pd.DataFrame(housing.data, columns=housing.feature_names)
7 df['MedHouseVal'] = housing.target
8
```

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 > ...
1  from sklearn.impute import SimpleImputer
2  from sklearn.preprocessing import StandardScaler
3
4  # Handle missing values
5  imputer = SimpleImputer(strategy='median')
6  df = pd.DataFrame(imputer.fit_transform(df), columns=df.columns)
7
8  # Scale features
9  scaler = StandardScaler()
10 df[df.columns] = scaler.fit_transform(df)
11 |
```

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 > ...
1 X = df.drop('MedHouseVal', axis=1)
2 y = df['MedHouseVal']
3
```

4. Model Selection

Select regression techniques such as Linear Regression or Decision Trees.

```
PythonTask.py > ...
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
3 from sklearn.tree import DecisionTreeRegressor
4
5 # Split data into training and testing sets
6 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
7
8 # Initialize models
9 linear_reg = LinearRegression()
10 tree_reg = DecisionTreeRegressor()
11
```

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

5. Model Training

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

```
1 # Train Linear Regression model
2 linear_reg.fit(X_train, y_train)
3
4 # Train Decision Tree model
5 tree_reg.fit(X_train, y_train)
6
```

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.

```
1 from sklearn.metrics import mean_squared_error
2
3 # Predict using both models
4 linear_pred = linear_reg.predict(X_test)
5 tree_pred = tree_reg.predict(X_test)
6
7 # Calculate MSE
8 linear_mse = mean_squared_error(y_test, linear_pred)
9 tree_mse = mean_squared_error(y_test, tree_pred)
10
11 print(f"Linear Regression MSE: {linear_mse}")
12 print(f"Decision Tree MSE: {tree_mse}")
13
```

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.

Task 02

Task 2



Classify Iris Flowers

Build a classification model using the Iris dataset to classify iris flowers into three species (Setosa, Versicolor, Virginica) based on their sepal and petal dimensions. Preprocess the dataset by splitting it into training and testing sets. Select a suitable classification algorithm such as Logistic Regression, Decision Trees, or SVM to train on the training data. Evaluate the model's accuracy and performance using metrics like accuracy, precision, recall, and F1-score on the test set to determine its effectiveness in classifying iris species. This model aims to provide accurate species classification based on flower measurements.

1. Load the Iris Dataset

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

```
PythonTask.py > ...  
1  from sklearn.datasets import load_iris  
2  import pandas as pd  
3  
4  # Load Iris dataset  
5  iris = load_iris()  
6  df = pd.DataFrame(iris.data, columns=iris.feature_names)  
7  df['species'] = iris.target  
8
```

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.

```
PythonTask.py > ...
1  from sklearn.model_selection import train_test_split
2
3  X = df.drop('species', axis=1)
4  y = df['species']
5
6  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
7
```

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

```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.svm import SVC
4
5 # Initialize classifiers
6 log_reg = LogisticRegression(max_iter=200)
7 tree_clf = DecisionTreeClassifier()
8 svm_clf = SVC()
9
```

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
1 # Train Logistic Regression model
2 log_reg.fit(X_train, y_train)
3
4 # Train Decision Tree model
5 tree_clf.fit(X_train, y_train)
6
7 # Train SVM model
8 svm_clf.fit(X_train, y_train)
9
```

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 > ...
1 from sklearn.metrics import accuracy_score, classification_report
2
3 # Predict using the models
4 log_reg_pred = log_reg.predict(X_test)
5 tree_clf_pred = tree_clf.predict(X_test)
6 svm_clf_pred = svm_clf.predict(X_test)
7
8 # Evaluate Logistic Regression
9 print("Logistic Regression Accuracy:", accuracy_score(y_test, log_reg_pred))
10 print("Logistic Regression Report:\n", classification_report(y_test, log_reg_pred, target_names=iris.target_names))
11
12 # Evaluate Decision Tree
13 print("Decision Tree Accuracy:", accuracy_score(y_test, tree_clf_pred))
14 print("Decision Tree Report:\n", classification_report(y_test, tree_clf_pred, target_names=iris.target_names))
15
16 # Evaluate SVM
17 print("SVM Accuracy:", accuracy_score(y_test, svm_clf_pred))
18 print("SVM Report:\n", classification_report(y_test, svm_clf_pred, target_names=iris.target_names))
19
```

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

Task 3



Image Classification

Build a deep learning model using the CIFAR-10 dataset to classify images into 10 distinct categories, including objects like airplanes, cars, and animals. Prepare the dataset by standardizing image sizes and normalizing pixel values. Select a convolutional neural network (CNN) architecture like ResNet or VGG for its effectiveness in image classification tasks. Train the model on labeled images to learn features that differentiate between different classes. Evaluate the model's accuracy and performance metrics on a separate test set to validate its capability in accurately categorizing images across varied classes.

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 task.py / ...
1 import tensorflow as tf
2 from tensorflow.keras import datasets, layers, models
3 import matplotlib.pyplot as plt
4 |
```

2. Load and Prepare the CIFAR-10 Dataset

```
Python task.py
1 (train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
2
3 # Normalize pixel values to be between 0 and 1
4 train_images, test_images = train_images / 255.0, test_images / 255.0
5
```

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

3. Visualize the Dataset (Optional)

```
1 class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
2
3 plt.figure(figsize=(10,10))
4 for i in range(25):
5     plt.subplot(5,5,i+1)
6     plt.xticks([])
7     plt.yticks([])
8     plt.grid(False)
9     plt.imshow(train_images[i], cmap=plt.cm.binary)
10    plt.xlabel(class_names[train_labels[i][0]])
11 plt.show()
12
13
```

4. Build the CNN Model

```
1 model = models.Sequential([
2     layers.Conv2D(32, (3,3), activation='relu', input_shape=(32, 32, 3)),
3     layers.MaxPooling2D((2, 2)),
4     layers.Conv2D(64, (3, 3), activation='relu'),
5     layers.MaxPooling2D((2, 2)),
6     layers.Conv2D(64, (3, 3), activation='relu'),
7     layers.Flatten(),
8     layers.Dense(64, activation='relu'),
9     layers.Dense(10)
10 ])
11
```

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

5. Compile the Model

```
1 model.compile(optimizer='adam',  
2               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  
3               metrics=['accuracy'])  
4
```

6. Train the Model

```
1 history = model.fit(train_images, train_labels, epochs=10,  
2                     validation_data=(test_images, test_labels))  
3
```

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

7. Evaluate the Model

```
Python3: 7...
1 test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
2 print(f'\nTest accuracy: {test_acc}')
3
4
```

8. Plot the Accuracy and Loss Over Time

```
Python3: 8...
1 plt.plot(history.history['accuracy'], label='accuracy')
2 plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
3 plt.xlabel('Epoch')
4 plt.ylabel('Accuracy')
5 plt.ylim([0, 1])
6 plt.legend(loc='lower right')
7 plt.show()
8
9
```

Name M Hammad Hussain

Email hammadhussain1972@gmail.com

Domain Machine learning

9. Make Predictions

```
1 probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
2 predictions = probability_model.predict(test_images)
3
4 # Example prediction
5 print(class_names[np.argmax(predictions[0])])
6
7
```

10. Save the Model

```
model.save('cifar10_model.h5')
```