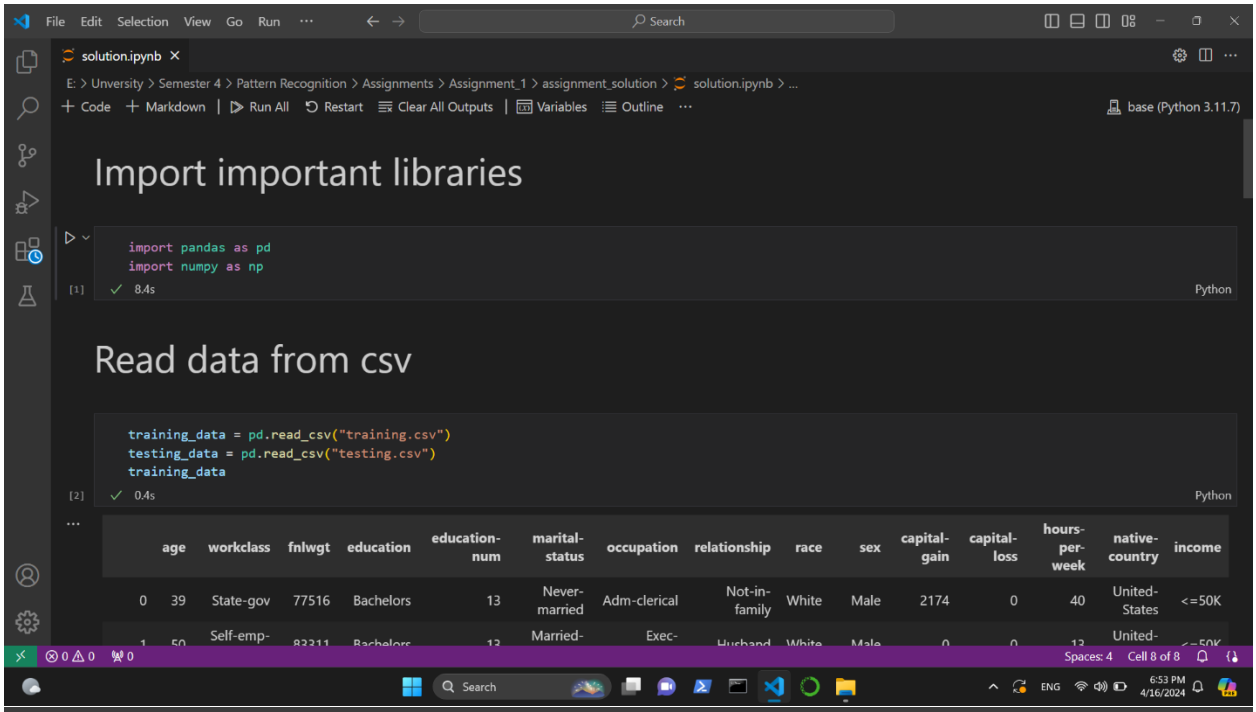
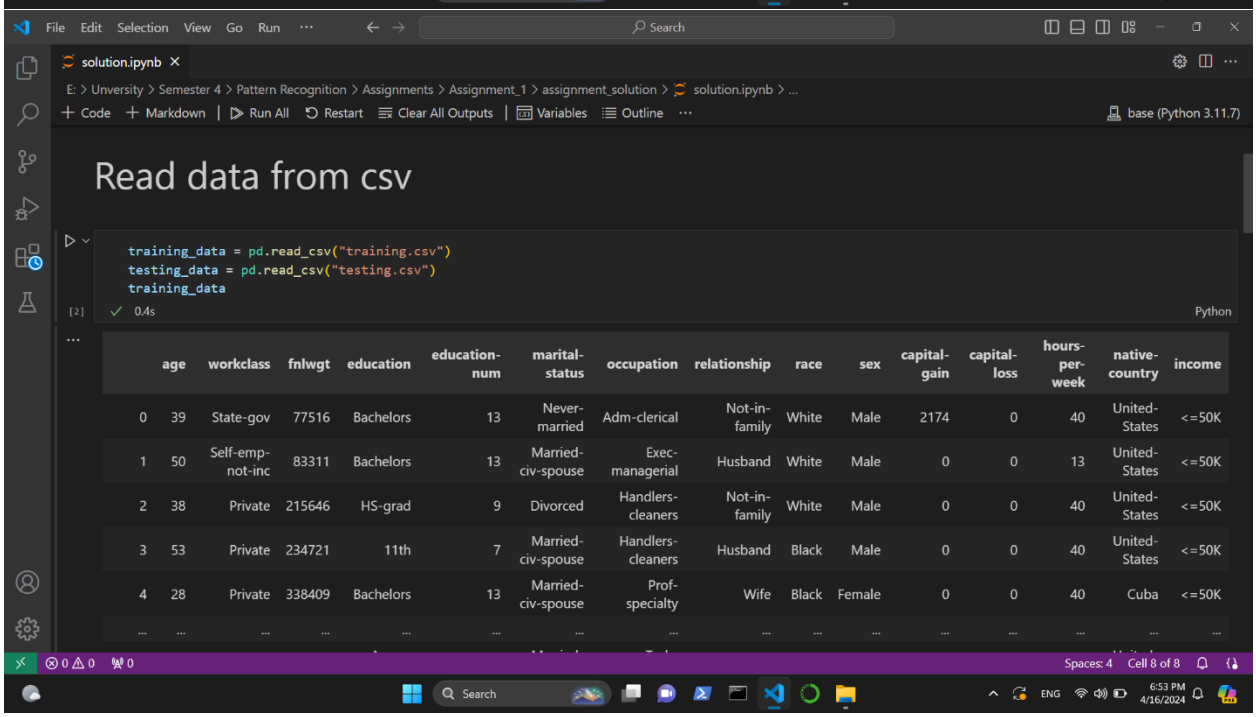
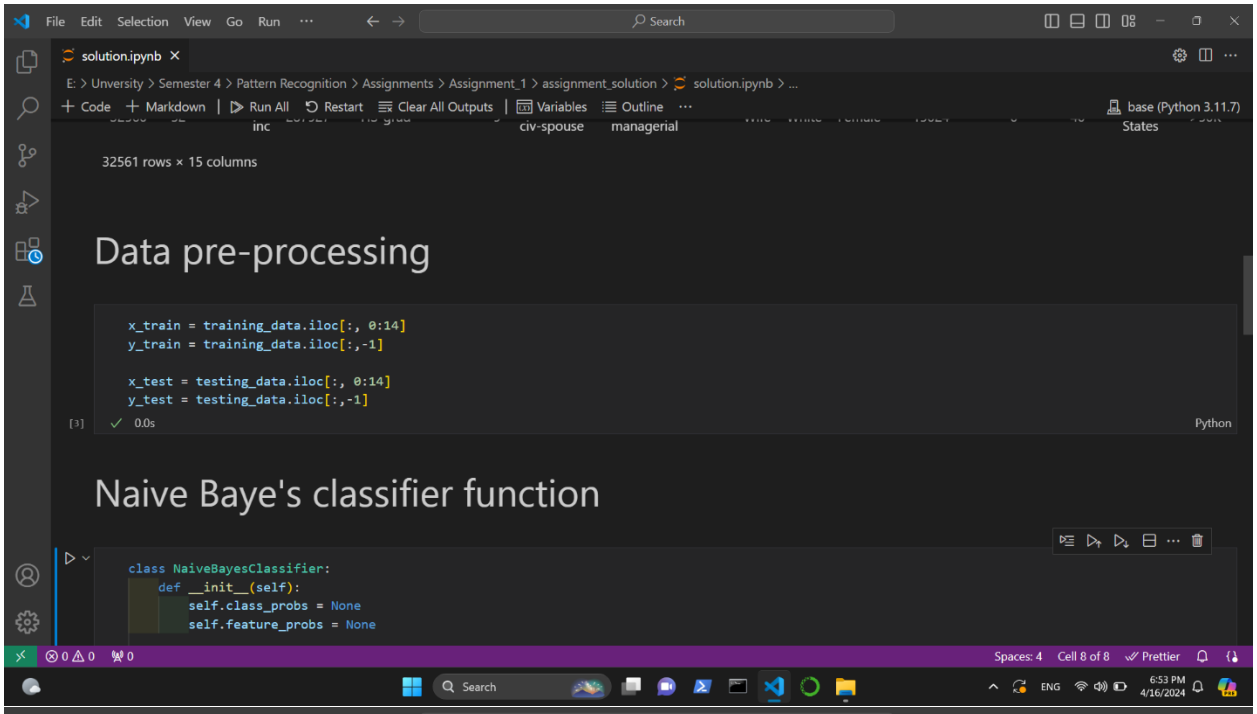
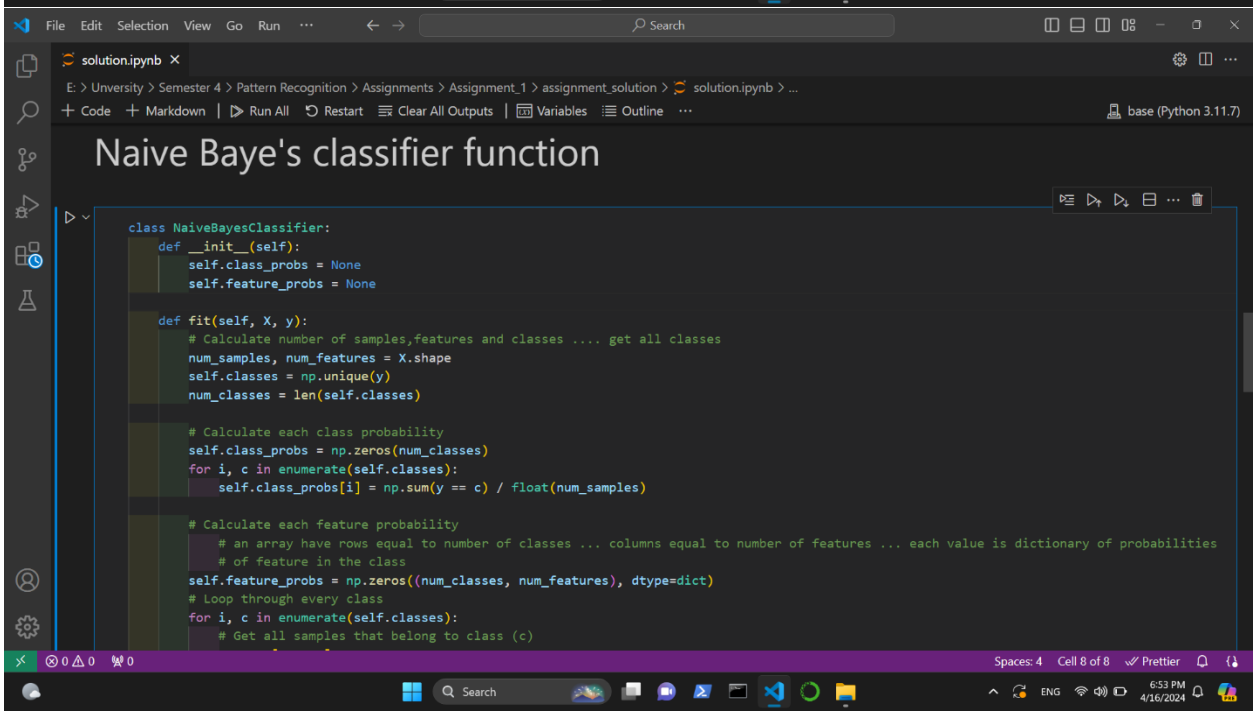


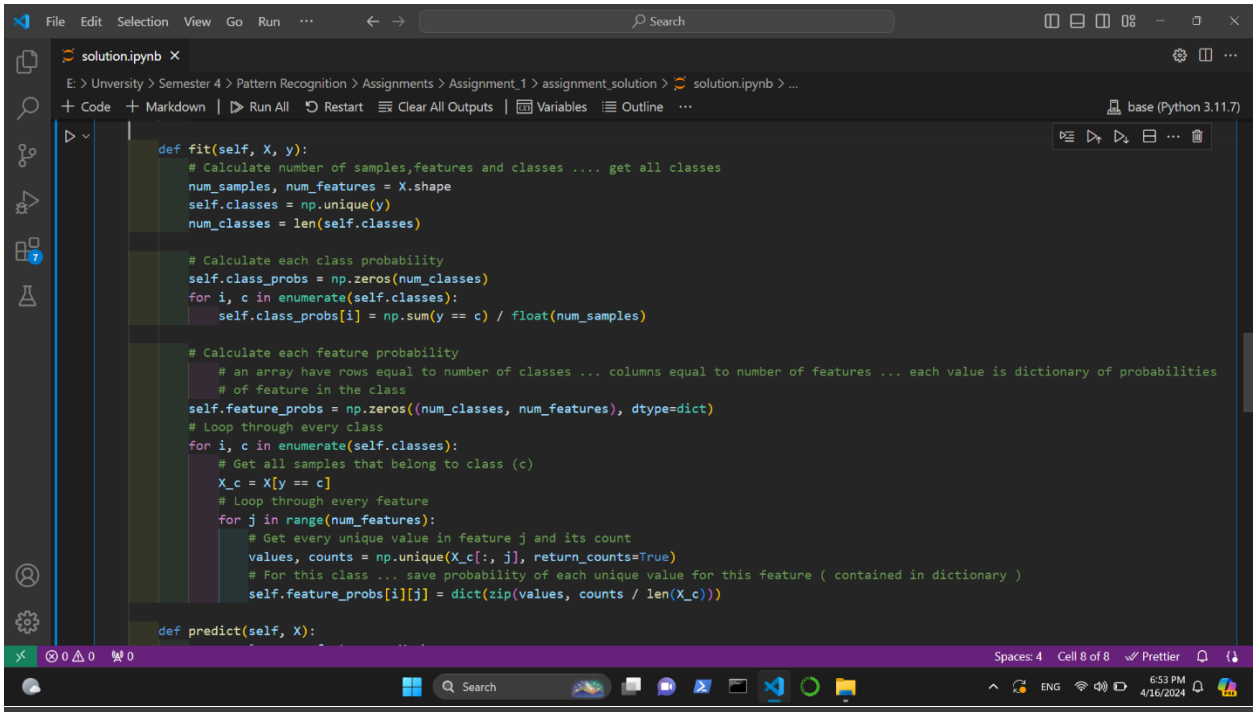
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K



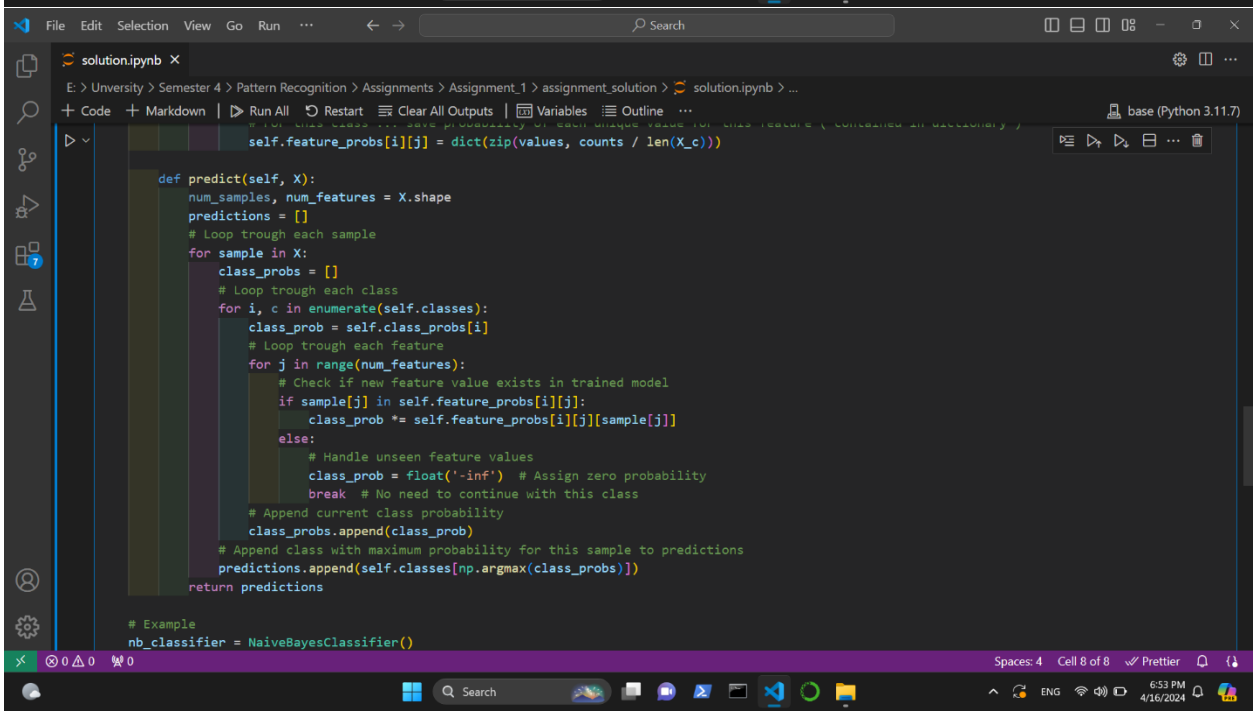
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K


File Edit Selection View Go Run
Search
solution.ipynb
E: > Unversity > Semester 4 > Pattern Recognition > Assignments > Assignment_1 > assignment_solution > solution.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.7)
inc
civ-spouse
managerial
States
32561 rows × 15 columns
Data pre-processing
x_train = training_data.iloc[:, 0:14]
y_train = training_data.iloc[:,-1]
x_test = testing_data.iloc[:, 0:14]
y_test = testing_data.iloc[:,-1]
[3] 0.0s
Python
Naive Baye's classifier function
class NaiveBayesClassifier:
def __init__(self):
self.class_probs = None
self.feature_probs = None
Spaces: 4 Cell 8 of 8 Prettier
Search
ENG
6:53 PM
4/16/2024

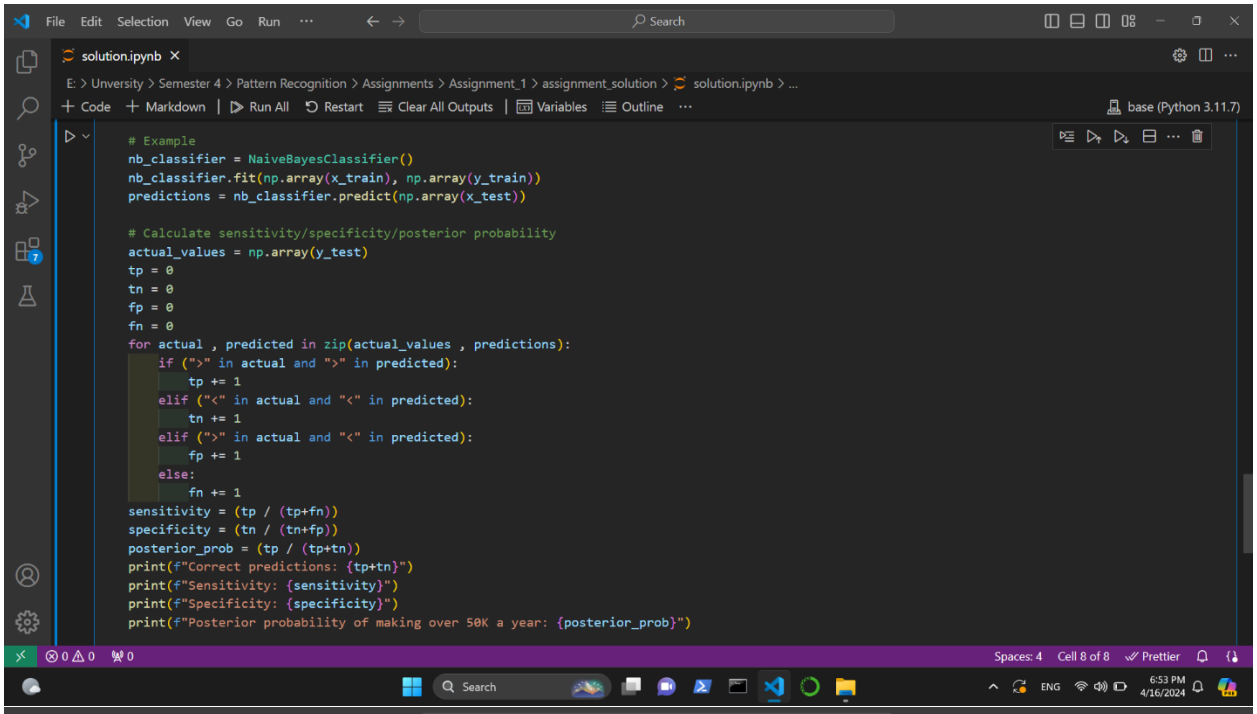

File Edit Selection View Go Run
Search
solution.ipynb
E: > Unversity > Semester 4 > Pattern Recognition > Assignments > Assignment_1 > assignment_solution > solution.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.7)
Naive Baye's classifier function
class NaiveBayesClassifier:
def __init__(self):
self.class_probs = None
self.feature_probs = None
def fit(self, X, y):
# Calculate number of samples,features and classes .... get all classes
num_samples, num_features = X.shape
self.classes = np.unique(y)
num_classes = len(self.classes)
# Calculate each class probability
self.class_probs = np.zeros(num_classes)
for i, c in enumerate(self.classes):
self.class_probs[i] = np.sum(y == c) / float(num_samples)
# Calculate each feature probability
# an array have rows equal to number of classes ... columns equal to number of features ... each value is dictionary of probabilities
# of feature in the class
self.feature_probs = np.zeros((num_classes, num_features), dtype=dict)
# Loop through every class
for i, c in enumerate(self.classes):
# Get all samples that belong to class (c)
Spaces: 4 Cell 8 of 8 Prettier
Search
ENG
6:53 PM
4/16/2024

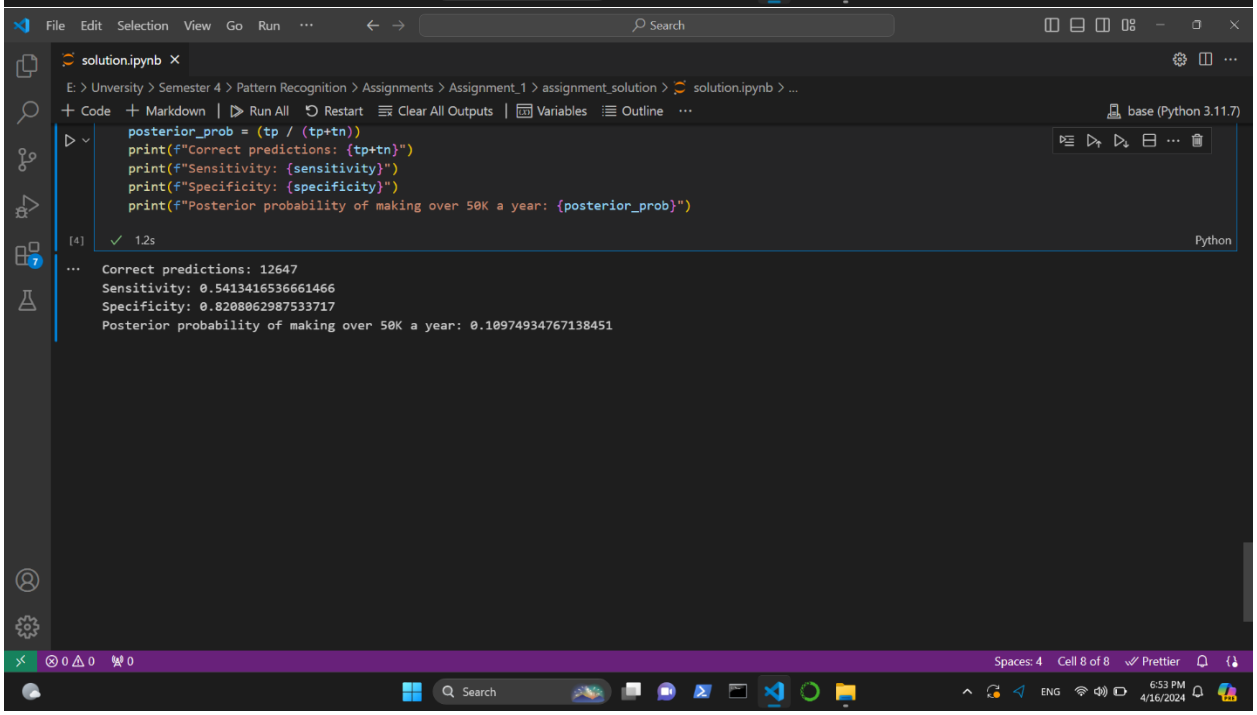


```
    def fit(self, X, y):
        # Calculate number of samples,features and classes .... get all classes
        num_samples, num_features = X.shape
        self.classes = np.unique(y)
        num_classes = len(self.classes)

        # Calculate each class probability
        self.class_probs = np.zeros(num_classes)
        for i, c in enumerate(self.classes):
            self.class_probs[i] = np.sum(y == c) / float(num_samples)

        # Calculate each feature probability
            # an array have rows equal to number of classes ... columns equal to number of features ... each value is dictionary of probabilities
            # of feature in the class
        self.feature_probs = np.zeros((num_classes, num_features), dtype=dict)
        # Loop through every class
        for i, c in enumerate(self.classes):
            # Get all samples that belong to class (c)
            X_c = X[y == c]
            # Loop through every feature
            for j in range(num_features):
                # Get every unique value in feature j and its count
                values, counts = np.unique(X_c[:, j], return_counts=True)
                # For this class ... save probability of each unique value for this feature ( contained in dictionary )
                self.feature_probs[i][j] = dict(zip(values, counts / len(X_c)))

    def predict(self, X):
```



```
                self.feature_probs[i][j] = dict(zip(values, counts / len(X_c)))

    def predict(self, X):
        num_samples, num_features = X.shape
        predictions = []
        # Loop trough each sample
        for sample in X:
            class_probs = []
            # Loop trough each class
            for i, c in enumerate(self.classes):
                class_prob = self.class_probs[i]
                # Loop trough each feature
                for j in range(num_features):
                    # Check if new feature value exists in trained model
                    if sample[j] in self.feature_probs[i][j]:
                        class_prob *= self.feature_probs[i][j][sample[j]]
                    else:
                        # Handle unseen feature values
                        class_prob = float('-inf')  # Assign zero probability
                        break  # No need to continue with this class
                # Append current class probability
                class_probs.append(class_prob)
            # Append class with maximum probability for this sample to predictions
            predictions.append(self.classes[np.argmax(class_probs)])
        return predictions

# Example
nb_classifier = NaiveBayesClassifier()
```


File Edit Selection View Go Run
Search
solution.ipynb
E: > Unversity > Semester 4 > Pattern Recognition > Assignments > Assignment_1 > assignment_solution > solution.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.7)
# Example
nb_classifier = NaiveBayesClassifier()
nb_classifier.fit(np.array(x_train), np.array(y_train))
predictions = nb_classifier.predict(np.array(x_test))
# Calculate sensitivity/specificity/posterior probability
actual_values = np.array(y_test)
tp = 0
tn = 0
fp = 0
fn = 0
for actual , predicted in zip(actual_values , predictions):
if (">" in actual and ">" in predicted):
tp += 1
elif ("<" in actual and "<" in predicted):
tn += 1
elif (">" in actual and "<" in predicted):
fp += 1
else:
fn += 1
sensitivity = (tp / (tp+fn))
specificity = (tn / (tn+fp))
posterior_prob = (tp / (tp+tn))
print(f"Correct predictions: {tp+tn}")
print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"Posterior probability of making over 50K a year: {posterior_prob}")
Spaces: 4 Cell 8 of 8 Prettier
Search
ENG
6:53 PM
4/16/2024


File Edit Selection View Go Run
Search
solution.ipynb
E: > Unversity > Semester 4 > Pattern Recognition > Assignments > Assignment_1 > assignment_solution > solution.ipynb > ...
Code Markdown Run All Restart Clear All Outputs Variables Outline
base (Python 3.11.7)
posterior_prob = (tp / (tp+tn))
print(f"Correct predictions: {tp+tn}")
print(f"Sensitivity: {sensitivity}")
print(f"Specificity: {specificity}")
print(f"Posterior probability of making over 50K a year: {posterior_prob}")
[4] 1.2s
Python
Correct predictions: 12647
Sensitivity: 0.5413416536661466
Specificity: 0.8208062987533717
Posterior probability of making over 50K a year: 0.10974934767138451
Spaces: 4 Cell 8 of 8 Prettier
Search
ENG
6:53 PM
4/16/2024