Step 1: Imported all the important libraries and also the data. Then had a look at the data to understand.

Step 2: Checked for any NaN values, and dtypes, shape, etc. This gave a brief outlook of the dataframe

Step 3: To reduce the classification classes, started clubbing the various suggested job profiles under a common domain like Analysts, Engineers, etc.

Step 4: At this point, the data frame has values both as integer and object, strings, etc. Now for a proper ML modelling, the values need to be in numeric format to be properly interpreted and predicted. So using Label-Encoding for the same. Using this, encoding the values under a column that were not numeric earlier, to a numeric value.

Step 5: Now we can start with splitting the data into 2 sets, one that will be used to train the model and another on which the model's prediction correctness will be tested. Since this process needs to be done 4-5 times for various train:Test ratio, making a function for the same.

Step 6: In the ANN_Model function, which is the main function, we split the data, then scale some out of bounds value for an even better optimized prediction model. Now using MLPClassifier, creating & training an ANN Model with the mentioned hidden layers and neurons. Then finding the accuracy score of the model, against the test data.

Step 7: Also creating a Confusion Matrix, and for each class, calculating the class-wise accuracy.

Step 8: Calling these functions on various train:test ratios and plotting them for a better visual analysis & interpreting the results. The best accuracy score occurs around 80:20 split ratio.

Step 9: Included the code to attach the Assignment 4 with Assignment 1. Modified the attributes of the given data with the courses used in Assignment 1 and since their marks are already present in the data, it can be used accordingly. Also, once again changing the career paths into the nearest possible electives careers, so that in a way it resembles and connects to the Assignment 1, as now it will take marks, electives, etc into input and then predict our desired result considering all these parameters. This was essentially the gist of Assignment 1 which is now successfully implemented/linked, as per instructions.

Also for this step, plotted a graph to check the effect of various courses on the output being predicted in the dataset. So that the model is as close to the original data and real world.

jupyter AI_A3 Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Markdown | Commit

```
df[colname] = label_encoder.fit_transform(df[colname])
label_encoding()
```

In [222]:
```
X = df.drop(columns=['Suggested Job Role'])
Y = df['Suggested Job Role']
```

In [223]:
```
def classwise_Accuracy(Confusion_Matr):
    for confusion_matrix in Confusion_Matr:
        diagonal_sum = confusion_matrix.trace()
        sum_of_all_elements = confusion_matrix.sum()
        print("Confusion Matrix", confusion_matrix)
        print("Its accuracy",diagonal_sum / sum_of_all_elements)
        print("\n")
```

In [224]:
```
score_trend=[]
testratio_trend=[]
def ANN_Model(testsize):
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=testsize, random_state=2002)
    sc_X = StandardScaler()
    X_train_sc= sc_X.fit_transform(X_train)
    X_test_sc= sc_X.transform(X_test)

    model = MLPClassifier(activation='relu', early_stopping= True, hidden_layer_sizes=(128,64,32), max_iter = 1000).fit(X_train_
    y_pred=model.predict(X_test)
    #y_pred=model.predict(entry_df)
    score=model.score(X_train_sc, Y_train)
    print("Accuracy score:",score)
    score_trend.append(score*100)
    testratio_trend.append(testsize*100)
    classwise_Accuracy(multilabel_confusion_matrix(Y_test, y_pred, labels= np.unique(Y)))
```

jupyter AI_A3 Last Checkpoint: 13 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Markdown | Commit

Train:Test -> 80:20

In [226]:
```
ANN_Model(0.2)
```

```
Accuracy score: 0.20475
Confusion Matrix [[3445    0]
 [ 555    0]]
Its accuracy 0.86125


Confusion Matrix [[   0 3232]
 [   0  768]]
Its accuracy 0.192


Confusion Matrix [[3770    0]
 [ 230    0]]
Its accuracy 0.9425


Confusion Matrix [[3766    0]
 [ 234    0]]
Its accuracy 0.9415


Confusion Matrix [[3282    0]
 [ 718    0]]
Its accuracy 0.8205


Confusion Matrix [[3406    0]
 [ 594    0]]
Its accuracy 0.8515


Confusion Matrix [[3666    0]
 [ 334    0]]
Its accuracy 0.9165


Confusion Matrix [[3668    0]
 [ 332    0]]
Its accuracy 0.917
```

jupyter AI_A3 Last Checkpoint: 14 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

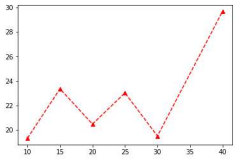
Trusted | Python 3

Run | Markdown | Commit

```
Confusion Matrix [[4575    0]
 [ 425    0]]
Its accuracy 0.915


Confusion Matrix [[4593    0]
 [ 407    0]]
Its accuracy 0.9186


Confusion Matrix [[4704    0]
 [ 296    0]]
Its accuracy 0.9408
```

In [231]:
```
import matplotlib.pyplot as plt

score_trend=[y for x, y in sorted(zip(testratio_trend, score_trend))]
plt.plot(sorted(testratio_trend),score_trend,'r^--')
print(testratio_trend)
```

```
[10.0, 20.0, 30.0, 40.0, 15.0, 25.0]
```



Modifying for connecting the data to Assignment 1, point g of Assignment