Job Selection Machine Learning Project

Authors
Harnoor Singh , Apurva Singh Prabhjee Singh

Link to Original Data Set
https://www.kaggle.com/datasets/suryasanju/interview-selection-dataset/

Link to Video
Interview Selection Machine Learning Project Presentation - YouTube

The problem and the big picture
For this machine learning project we used supervised learning as we have a dataset with training examples labeled. This is a classification task as we are predicting a state (whether a candidate will get a job offer or not.) Finally the technique used is batch learning as we don't have a continuous flow of data into our system .

Our project will predict whether the candidate is offered a job after the interview has been conducted. We also analyzed which factors were most significant in determining a candidate's success in getting a job offer.

Interview Selection Dataset Description
The dataset has 52 attributes , listed below

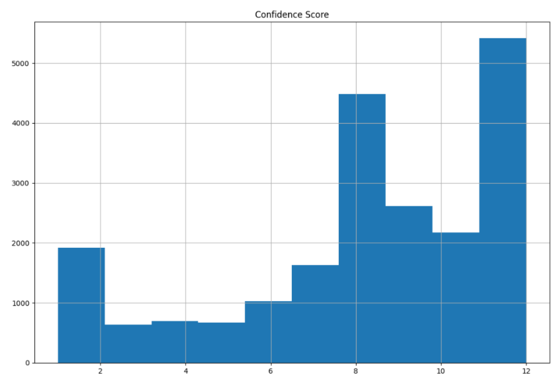
1. Name
2. Age
3. Gender
4. Type of Graduation/Post Graduation
5. Marital status Mode of interview given by candidate?
6. Pre Interview Check
7. Fluency in English based on introduction
8. Confidence based on Introduction (English)
9. Confidence based on the topic given
10. Confidence Based on the PPT Question
11. Confidence based on the sales scenario
12. Structured Thinking (In regional only)
13. Structured Thinking Based on the PPT Question
14. Structured Thinking(Call pitch)
15. Regional fluency based on the topic given
16. Regional fluency Based on the PPT Question
17. Regional fluency based on the sales scenario
18. Does the candidate has mother tongue influence while speaking english.
19. Has acquaintance in Company and has spoken to him/her before applying?
20. Candidate Status
21. Last Fixed CTC (lakhs)
22. Currently Employed
23. Experienced candidate - (Experience in months)
24. Experienced Candidate (Nature of work)
25. "What was the type of Role? "
26. How many slides candidate have submitted in PPT?
27. Call-pitch Elements used during the call Sales Scenario
28. But, my child's exam are going on now, so we will keep the counselling session after the exams get over.(Time: Favourable pitch: Counsellor hype)
29. Let me discuss it with my child
30. Sir being in education industry I know this is a marketing gimmick and at the end of the day you'll be selling the app.
31. Role acceptance
32. Interview Verdict
33. Candidate is willing to relocate
34. Role Location to be given to the candidate
35. Comments
36. RedFlags Comments in Interview
37. Confidence based on Introduction (English)
38. Confidence based on the topic given
39. Confidence Based on the PPT Question
40. Confidence based on the sales scenario
41. Structured Thinking (In regional only)
42. Structured Thinking Based on the PPT Question
43. Structured Thinking(Call pitch)
44. Regional fluency based on the topic given

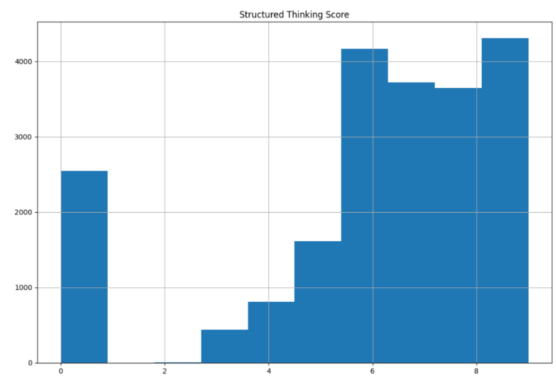
45. Regional fluency Based on the PPT Question
46. Regional fluency based on the sales scenario
47. Confidence Score
48. Structured Thinking Score
49. Regional Fluency Score
50. Total Score
51. "Whether joined the company or not

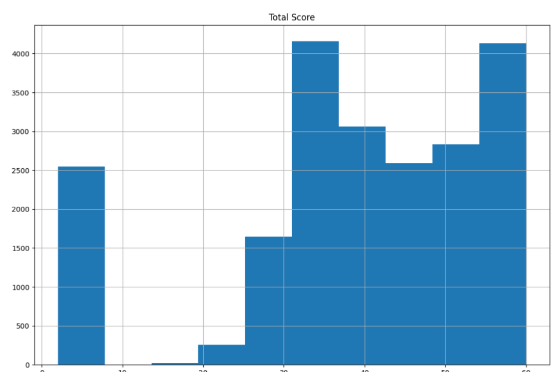
We removed 25 of the following attributes from the dataset:

Name', 'Gender',   'Marital status', 'Mode of interview given by candidate?', Pre Interview Check', 'Confidence based on Introduction (English)', 'Confidence based on the topic given’,   'Confidence Based on the PPT Question', 'Confidence based on the sales scenario','Structured Thinking (In regional only)', 'Structured Thinking Based on the PPT Question',
 'Structured Thinking(Call pitch)', 'Regional fluency based on the topic given ', 'Regional fluency Based on the PPT Question','Regional fluency based on the sales scenario','Has acquaintance in Company and has spoken to him/her before applying?', 'Last Fixed CTC (lakhs) ', 'Currently Employed', 'How many slides candidate have submitted in PPT?',
 'Role Location to be given to the candidate', 'Comments', 'RedFlags Comments in Interview', 'Whether joined the company or not\n'
, 'Role acceptance', 'Experienced Candidate (Nature of work)

Graphs







The first graph is the confidence score graph and the second graph is the structured thinking score. After looking at the 3 graphs we can hypothesize that Structured thinking could be a major factor in whether a candidate gets hired or not. The graph for Structured thinking has many incidences of a low score whereas the confidence graph has a more even distribution of incidences for all scores.

Data cleaning and preprocessing.

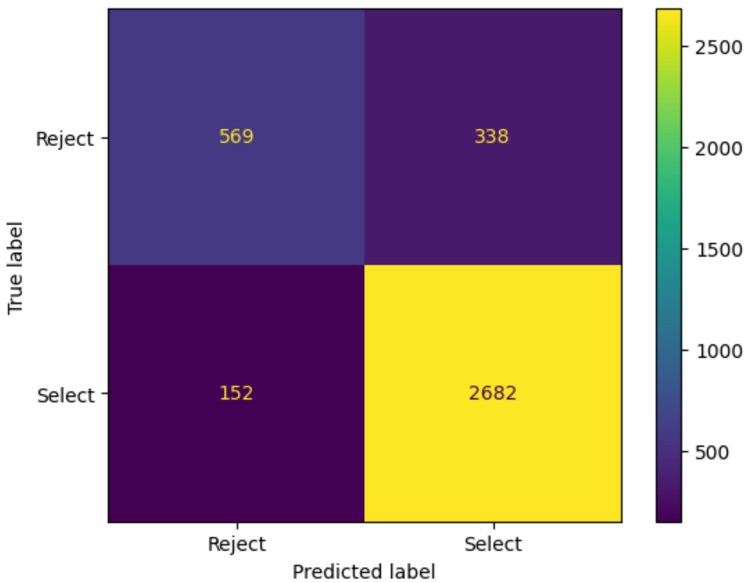
During our Data Cleaning we removed duplicates, empty values , and dropped redundant columns. There were certain columns that had already been translated into numerical data , however the dataset has included both. We removed such columns as well.

Algorithms and our Findings

We used 3 classification algorithms for this Project.
The first algorithm we used was **Random Forest Classification**. The results are below.

	precision	recall	f1-score	support
Reject	0.79	0.63	0.70	907
Select	0.89	0.95	0.92	2834
accuracy			0.87	3741
macro avg	0.84	0.79	0.81	3741
weighted avg	0.86	0.87	0.86	3741

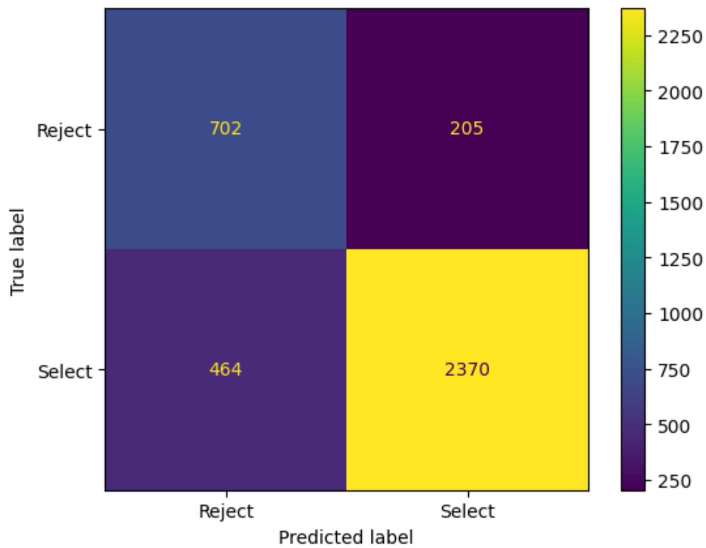
And the confusion matrix is below:



This algorithm performed well and we have a high f1-score, precision, recall for “Select”.

The second Algorithm we used was **Naives Bayes Gaussian**. The results are below:

	precision	recall	f1-score	support
Reject	0.60	0.77	0.68	907
Select	0.92	0.84	0.88	2834
accuracy			0.82	3741
macro avg	0.76	0.81	0.78	3741
weighted avg	0.84	0.82	0.83	3741



This algorithm is not as great at RFC at predicting Reject values, however it has a high precision in regards to Correctly predicting Select.

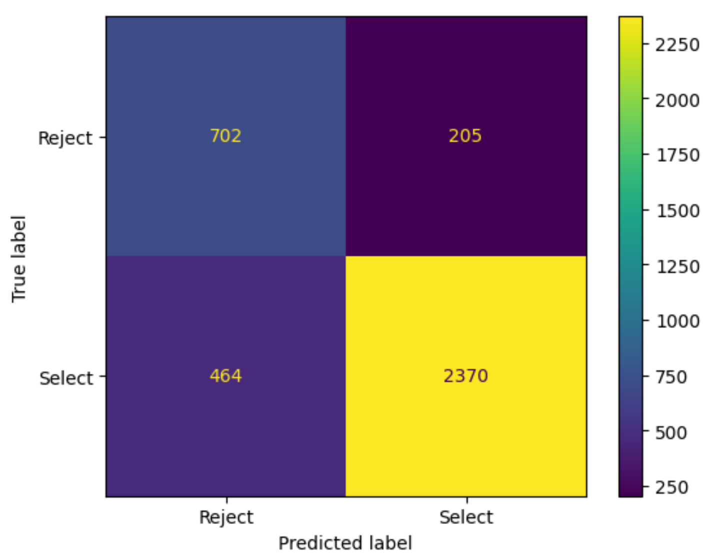
The third algorithm we used was the Linear **algorithm.** The classification report is below.

```
              precision    recall  f1-score   support

      Reject       0.60      0.77      0.68       907
      Select       0.92      0.84      0.88      2834

    accuracy                           0.82      3741
   macro avg       0.76      0.81      0.78      3741
weighted avg       0.84      0.82      0.83      3741
```

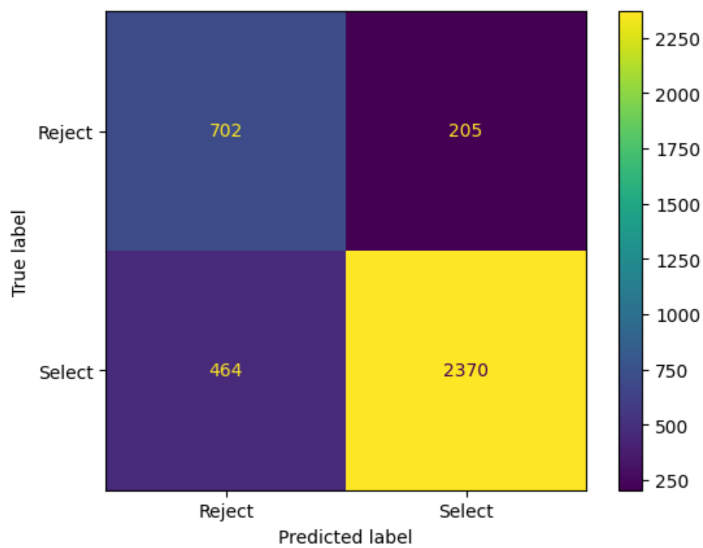
The confusion matrix is below.

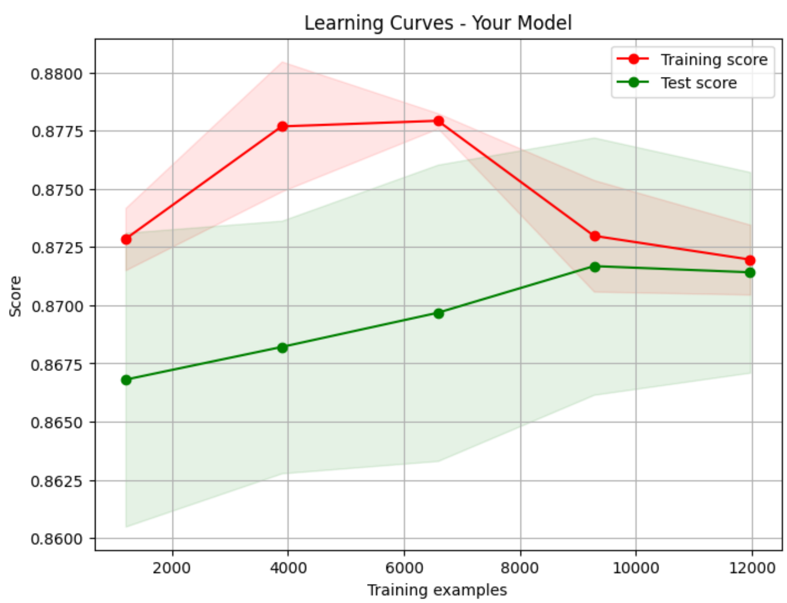


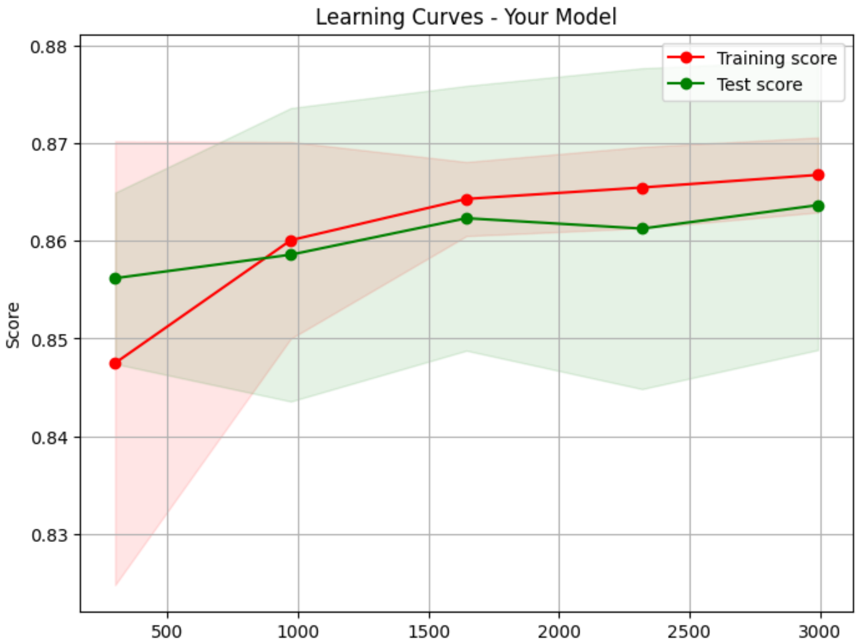
This algorithm performed on par with the Random Forest Algorithm. However it seems to have better numbers for recall for Reject.

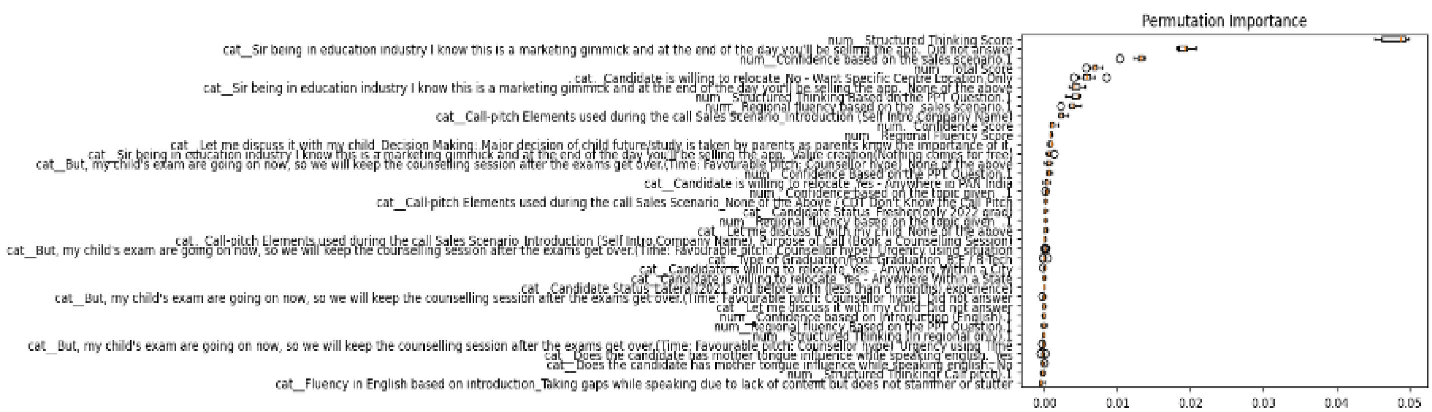
After looking at the classification reports and confusion matrices for all 3 algorithms we concluded that the best algorithm was the Linear algorithm.

3 graphs for the best performing algorithm(Linear).









We picked the Linear model as the best performing algorithm. Although all three algorithms did certain things well, The Linear Algorithm has the best overall performance for both SELECT and “REJECT”

We checked our model to make sure there was no overfitting. Since the lines in the graphs above are converging as the data increases we can conclude that our feature selection is correct and model is not overfitting

The linear Model had the best combination of metrics for both Reject and Select, which we believed would give a more rounded picture.

As you can see from the permutation importance graph, the most significant factor in the final result was the structured thinking score.

Limitations

One of the major Limitations we faced was the bias in data . We had more data for candidates that were selected vs rejected, therefore our metrics for Reject would not be as accurate.

Appendix 1

Importing Required Libraries

```
import opendatasets as od
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Getting the Data

```
# Getting the dataset from the github repo

url = "https://raw.githubusercontent.com/PrabhjeeSingh/MLInterview-Selection-Project/main/DatasetFromKaggle.csv"
```

Defining DataFrame

```
data_df = pd.read_csv(url,on_bad_lines='skip')

# displaying the contents of the XLSX file
 data_df.head()
```

3 Graph of EDA

```
 columns = ["Confidence Score", "Structured Thinking Score", "Total Score"]
 data_df[columns].hist(figsize=(30, 20))
 # data_df.hist(figsize=(30,20))
 plt.show()
```

Columns in the dataset

In [6]:

```
data_df.columns
```

These columns doesn't add any value to the model

In [7]:

```
# Name, Gender, Marital Status, Mode of Interview, Pre interview check, Last CTC, Currently Employed, Whether joined,
# No. of slides in the presentation doesn't really reflect anything to the candidate selection.

# Rest of the column has already been quantitized from the qualitative comments by the dataset owner.

column_to_drop = [
        'Name',
        'Gender',
```

```
    'Marital status', 'Mode of interview given by candidate?',
    'Pre Interview Check',
    'Confidence based on Introduction (English)',
    'Confidence based on the topic given  ',
    'Confidence Based on the PPT Question',
    'Confidence based on the sales scenario',
    'Structured Thinking (In regional only)',
    'Structured Thinking Based on the PPT Question',
    'Structured Thinking( Call pitch)',
    'Regional fluency based on the topic given  ',
    'Regional fluency Based on the PPT Question',
    'Regional fluency based on the  sales scenario',
    'Has acquaintance in Company and has spoken to him/her before applying?',
    'Last Fixed CTC (lakhs) ', 'Currently Employed',
    'How many slides candidate have submitted in PPT?',
    'Role Location to be given to the candidate',
    'Comments',
    'RedFlags Comments in Interview',
    'Whether joined the company or not\n'   ,
  'Role acceptance',
  'Experienced Candidate (Nature of work)'
]
```

Preparing the Data

Dropping the redunctant columns in the dataset

```
# Dropping the columns
data_df = data_df.drop(columns=column_to_drop)
```

Columns we are considering from our dataset

```
data_df.columns

data_df.head()
```

Removing Empty Values

```
# Check for Empty Values
data_df.isna().sum()

# Delete Empty values
data_df.dropna(subset=["Interview Verdict"], inplace=True)
```

Duplicate Values

```
# Calculate Duplicate Values
```

```
data_df.duplicated().sum()
 data_df.drop_duplicates()
```

Configuring the values of Interview Verdict

```
data_df['Interview Verdict'].value_counts()
```

Combining the final result to the interview selection or rejection

```
data_df.loc[:,'Interview Verdict']=data_df['Interview Verdict'].replace({"Borderline Select":"Select",
                                                         "Premium Select":"Select",
                                                         "Borderline Reject":"Reject" })
```

```
data_df['Interview Verdict'].value_counts()
```

Data Pipeline Prepration

In [72]:

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
```

In [73]:

```
num_cols = data_df.select_dtypes(include='number').columns.to_list()
cat_cols = data_df.select_dtypes(exclude='number').columns.to_list()



# Exclude the target from numerical columns
cat_cols.remove("Interview Verdict")

# Create pipelines for numeric and categorical columns
num_pipeline = make_pipeline(SimpleImputer(strategy='mean'), StandardScaler())
cat_pipeline = make_pipeline(SimpleImputer(strategy='most_frequent'), OneHotEncoder(sparse_output=False))

# Use ColumnTransformer to set the estimators and transformations

preprocessing = ColumnTransformer([('num', num_pipeline, num_cols),
                                  ('cat', cat_pipeline, cat_cols)],
```

```
                    remainder='passthrough'
                   )
```

Numerical Columns

In [74]:

```
num_cols
```

Categorical Columns

In [75]:

```
cat_cols
```

Starting the preprocessing

In [76]:

```
preprocessing
```

Applying the pipline

In [77]:

```
data_prepared = preprocessing.fit_transform(data_df)
feature_names=preprocessing.get_feature_names_out()
data_prepared = pd.DataFrame(data=data_prepared, columns=feature_names)

data_prepared
```

Prepared Columns

In [78]:

```
data_prepared.columns
```

Splitting the dataset

In [79]:

```
from sklearn.model_selection import train_test_split

X = data_prepared.drop(["remainder__Interview Verdict"],axis=1)
y = data_prepared["remainder__Interview Verdict"]

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

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

Feature Importance Comparison

Because our data had most categorical values so corr was ineffective

In [80]:

```
from sklearn.ensemble import RandomForestClassifier
```

```
import pandas as pd

# Assuming X_train, X_test, y_train, y_test are already defined

rfm = RandomForestClassifier(n_estimators=70, oob_score=True, n_jobs=-1,
                             random_state=101, max_features=None, min_samples_leaf=30)
rfm.fit(X_train, y_train)

# Get feature importances
feature_importances = rfm.feature_importances_

# Pair feature importances with corresponding column names
feature_importance_df = pd.DataFrame({'Feature': X_train.columns, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
 print(feature_importance_df)
```

Selecting the only features based on the Importance with the Threshold of 0.001

In [100]:

```
# Assuming X_train, X_test, y_train, y_test are already defined
# Assuming you have already fitted the RandomForestClassifier and obtained feature importances

# Set a threshold for feature importance
threshold = 0.001  # You can adjust this threshold as per your requirements

# Filter features based on the threshold
selected_features = feature_importance_df[feature_importance_df['Importance'] >= threshold]['Feature'].tolist()

# Retain only the selected features in your dataset
X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]

# Use the selected features for further analysis or modeling
```

In [82]:

```
 X_train_selected
```

Training and Evaluating 3 ML models

Model 1 (Random Forest Classifier)

In [83]:

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, ConfusionMatrixDisplay
```

In [84]:

```
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier (n_estimators=70, oob_score=True, n_jobs=-1,
random_state=101, max_features = None, min_samples_leaf = 30)
rfm. fit(X_train_selected, y_train)
y_predict=rfm.predict(X_test_selected)

 print (classification_report(y_test, y_predict))

ConfusionMatrixDisplay.from_predictions(y_test, y_predict)
```

Model 2 (Naive Bayes Gaussian)

In [86]:

```
from sklearn.naive_bayes import GaussianNB

nb= GaussianNB()
# nb.fit(X_train,y_train)
# y_predict = nb.predict(X_test)
nb.fit(X_train_selected,y_train)
y_predict = nb.predict(X_test_selected)

 print (classification_report(y_test, y_predict))

ConfusionMatrixDisplay.from_predictions(y_test, y_predict)
```

Model 3 (Linear)

In [88]:

```
model_svm = SVC(kernel='linear', C=0.1, gamma=0.1)
```

In [89]:

```
# model_svm.fit(X_train, y_train).values.ravel()
model_svm.fit(X_train_selected, y_train.values.ravel())
```

Out[89]:

```
SVC(C=0.1, gamma=0.1, kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [90]:

```
y_predict=model_svm.predict(X_test_selected)
print(f'classification_report for C = 0.1')
```

```
print (classification_report(y_test, y_predict))
```

Plotting 3 graphs for our best performing algorithm: Linear

In [91]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_predict)
```

Checking if our best performing model is overfitting

In [92]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve

# Function to plot learning curves
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=None, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure(figsize=(8, 6))
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")

    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Test score")

    plt.legend(loc="best")
```

```
    return plt


title = "Learning Curves - Your Model"
plot_learning_curve(model_svm, title, X_train_selected, y_train, cv=5, n_jobs=-1)

plt.show()
```

Since the line seems to be converging as the data increases we can conclude that our feature selection is correct and model is not overfitting

In [93]:

```
## Comparing on the test data for overfitting


plot_learning_curve(model_svm, title, X_test_selected, y_test, cv=5, n_jobs=-1)


plt.show()


import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [95]:

```
importances_svm = model_svm.coef_[0]
```

In [96]:

```
acc = accuracy_score(y_test, y_predict)
```

In [101]:

```
from sklearn.inspection import permutation_importance
import matplotlib.pyplot as plt
import numpy as np

# Perform permutation importance
result = permutation_importance(rfm, X_train_selected, y_train, n_repeats=10, random_state=42, n_jobs=2)
sorted_idx = result.importances_mean.argsort()

# Display a bar plot of the feature importance
fig, ax = plt.subplots()
```

```
ax.boxplot(result.importances[sorted_idx].T, vert=False, labels=X_train_selected.columns[sorted_idx])
ax.set_title("Permutation Importance")
fig.tight_layout()
plt.show()
```

Appendix 2:

Link to Video
[(48) Interview Selection Machine Learning Project Presentation - YouTube]

Link to Data-Set
https://www.kaggle.com/datasets/suryasanju/interview-selection-dataset/

Link to Notebook
https://github.com/PrabhjeeSingh/MLInterview-Selection-Project/blob/main/main.ipynb