Importing required libraries

In [140…

```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# to scale the data using z-score
from sklearn.preprocessing import StandardScaler

# to perform Logistic Regression, Linear Discriminant Analysis and CART (Decision T
import statsmodels.api as sm
import statsmodels.stats.api as sms
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import roc_auc_score,roc_curve,classification_report,confusion
from sklearn import metrics
import scipy.stats as stats

# To check model performance
from sklearn.metrics import mean_absolute_error, mean_squared_error

# to suppress warnings
import warnings

warnings.filterwarnings("ignore")
```

Problem Statement:

Context:

In your role as a statistician at the Republic of Indonesia Ministry of Health, you have been entrusted with a dataset containing information from a Contraceptive Prevalence Survey. This dataset encompasses data from 1473 married females who were either not pregnant or were uncertain of their pregnancy status during the survey.

Your task involves predicting whether these women opt for a contraceptive method of choice. This prediction will be based on a comprehensive analysis of their demographic and socio-economic attributes.

Data Dictionary

1. Wife's age (numerical)

2. Wife's education (categorical) 1=uneducated, 2, 3, 4=tertiary
3. Husband's education (categorical) 1=uneducated, 2, 3, 4=tertiary
4. Number of children ever born (numerical)
5. Wife's religion (binary) Non-Scientology, Scientology
6. Wife's now working? (binary) Yes, No
7. Husband's occupation (categorical) 1, 2, 3, 4(random)
8. Standard-of-living index (categorical) 1=very low, 2, 3, 4=high
9. Media exposure (binary) Good, Not good
10. Contraceptive method used (class attribute) No,Yes

Understanding the structure of data

In [141…
```
df_lr = pd.read_excel('Contraceptive_method_dataset.xlsx')
```

In [142…
```
df_lr.head() # Returns first 5 rows
```

Out[142…

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife
0	24.0	Primary	Secondary	3.0	Scientology	
1	45.0	Uneducated	Secondary	10.0	Scientology	
2	43.0	Primary	Secondary	7.0	Scientology	
3	42.0	Secondary	Primary	9.0	Scientology	
4	36.0	Secondary	Secondary	8.0	Scientology	

Number of rows and columns in the dataset

In [143…
```
# checking shape of the data

rows = str(df_lr.shape[0])
columns = str(df_lr.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 1473 rows and 10 columns in the dataset.
```

Datatypes of the different columns in the dataset

In [144…
```
df_lr.info() # Concise summary of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1473 entries, 0 to 1472
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Wife_age                   1402 non-null   float64
 1   Wife_education             1473 non-null   object 
 2   Husband_education          1473 non-null   object 
 3   No_of_children_born        1452 non-null   float64
 4   Wife_religion              1473 non-null   object 
 5   Wife_Working               1473 non-null   object 
 6   Husband_Occupation         1473 non-null   int64  
 7   Standard_of_living_index   1473 non-null   object 
 8   Media_exposure             1473 non-null   object 
 9   Contraceptive_method_used  1473 non-null   object 
dtypes: float64(2), int64(1), object(7)
memory usage: 115.2+ KB
```

There are 10 columns in the dataset. Out of which 2 have float data type, 1 have integer data type and 7 have object data type.

Finding missing values in the dataset

In [145...
```
df_lr.isna().sum()  # Count NaN values in all columns of dataset
```

Out[145...
```
Wife_age                     71
Wife_education                0
Husband_education             0
No_of_children_born          21
Wife_religion                 0
Wife_Working                  0
Husband_Occupation            0
Standard_of_living_index      0
Media_exposure                0
Contraceptive_method_used     0
dtype: int64
```

Wife_age and No_of_children_born columns have NaN values in 71 and 21 rows.

Checking for Duplicates

In [146...
```
df_lr.duplicated().sum()
```

Out[146...
```
80
```

There are 80 duplicate rows in the dataset.

In [147...
```
df_lr.drop_duplicates(inplace=True)
```

In [148...
```
dups = df_lr.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 0
```

In [149…

```python
# checking shape of the data

rows = str(df_lr.shape[0])
columns = str(df_lr.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 1393 rows and 10 columns in the dataset (after duplicate rows removal).
```

Treating missing values in the dataset

In [150…

```python
# Use of fillna method to treat missing values in rchar and wchar columns

df_lr['Wife_age'] = df_lr['Wife_age'].fillna(df_lr['Wife_age'].median()) # Replace
df_lr['No_of_children_born'] = df_lr['No_of_children_born'].fillna(df_lr['No_of_chi
```

Median is used for treating the missing values for Wife_age and No_of_children_born columns as distribution is skewed.

In [151…

```python
df_lr.isna().sum()  # Count NaN values in all columns of dataset
```

Out[151…

```
Wife_age                     0
Wife_education               0
Husband_education            0
No_of_children_born          0
Wife_religion                0
Wife_Working                 0
Husband_Occupation           0
Standard_of_living_index     0
Media_exposure               0
Contraceptive_method_used    0
dtype: int64
```

We can see from above list that there are no NaN values in Wife_age and No_of_children_born columns.

Checking Summary Statistic

In [152…

```python
df_lr.describe(include='all').T
```

Out[152...

	count	unique	top	freq	mean	std	min	2!
Wife_age	1393.0	NaN	NaN	NaN	32.53051	8.088188	16.0	2
Wife_education	1393	4	Tertiary	515	NaN	NaN	NaN	N
Husband_education	1393	4	Tertiary	827	NaN	NaN	NaN	N
No_of_children_born	1393.0	NaN	NaN	NaN	3.286432	2.381791	0.0	
Wife_religion	1393	2	Scientology	1186	NaN	NaN	NaN	N
Wife_Working	1393	2	No	1043	NaN	NaN	NaN	N
Husband_Occupation	1393.0	NaN	NaN	NaN	2.174444	0.85459	1.0	
Standard_of_living_index	1393	4	Very High	618	NaN	NaN	NaN	N
Media_exposure	1393	2	Exposed	1284	NaN	NaN	NaN	N
Contraceptive_method_used	1393	2	Yes	779	NaN	NaN	NaN	N

Observations and Insights:

1. Average age of wife is 32 years (minimum - 16 years, maximum - 49 years).
2. Most wife and husband in a house are having Tertiary education.
3. Minimum and maximum number of children's are 0 and 16 in a house.
4. Most wife religion is Scientology in a house.
5. Most wife are not working in a house.
6. Most wife are exposed to media in a house.
7. Most house are having standard of living index as very high.

Categorial variables in the dataset

In [153...

```
df_lr['Wife_education'].value_counts().sort_values() # Frequency of each distinct v
```

Out[153...

```
Wife_education
Uneducated    150
Primary       330
Secondary     398
Tertiary      515
Name: count, dtype: int64
```

There are 4 levels of wife education (Uneducated, Primary, Secondary and Tertiary) with Tertiary having the maximum count.

In [154...

```
df_lr['Husband_education'].value_counts().sort_values() # Frequency of each distinc
```

Out[154…
```
Husband_education
Uneducated     44
Primary       175
Secondary     347
Tertiary      827
Name: count, dtype: int64
```

There are 4 levels of husband education (Uneducated, Primary, Secondary and Tertiary) with Tertiary having the maximum count.

In [155…
```
df_lr['Wife_religion'].value_counts().sort_values() # Frequency of each distinct va
```

Out[155…
```
Wife_religion
Non-Scientology     207
Scientology        1186
Name: count, dtype: int64
```

Wife's religion is Non-Scientology or Scientology. Scientology is having the maximum count.

In [156…
```
df_lr['Wife_Working'].value_counts().sort_values() # Frequency of each distinct val
```

Out[156…
```
Wife_Working
Yes     350
No     1043
Name: count, dtype: int64
```

Wife is either working or not working in a house. Not working is having the maximum count.

In [157…
```
df_lr['Standard_of_living_index'].value_counts().sort_values() # Frequency of each
```

Out[157…
```
Standard_of_living_index
Very Low     129
Low          227
High         419
Very High    618
Name: count, dtype: int64
```

Standard of living index in a house varies between Very Low, Low, High and Very High. Very High is having the maximum count.

In [158…
```
df_lr['Media_exposure'].value_counts().sort_values() # Frequency of each distinct v
```

Out[158…
```
Media_exposure
Not-Exposed     109
Exposed        1284
Name: count, dtype: int64
```

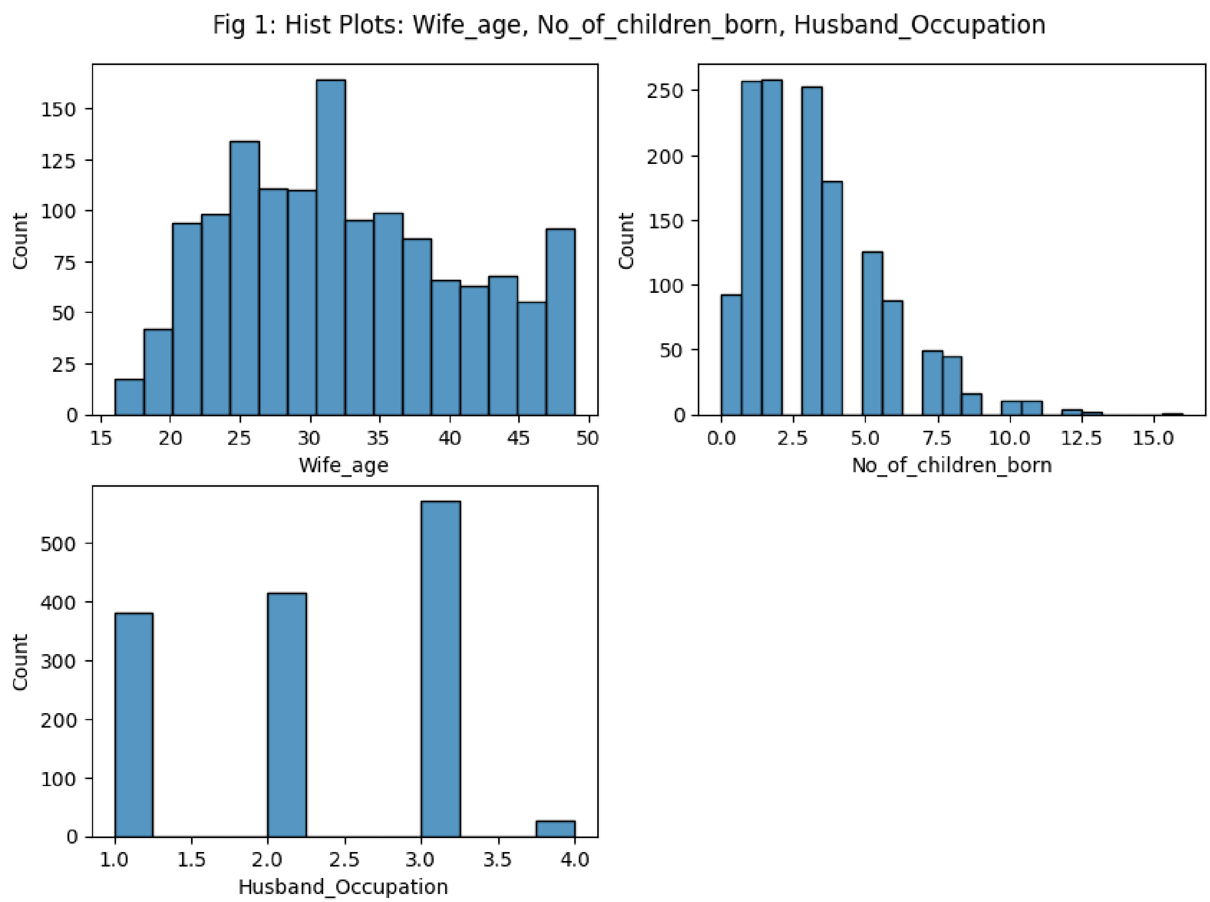
Wife is either Exposed or Not-Exposed to media. Exposed is having the maximum count.

Exploratory Data Analysis (EDA)

Univariate analysis

In [159…

```
# Hist Plots for Wife_age, No_of_children_born and Husband_Occupation

fig, axes = plt.subplots(2,2, figsize=(10, 7))

sns.histplot(ax=axes[0, 0], data=df_lr, x='Wife_age')
sns.histplot(ax=axes[0, 1], data=df_lr, x='No_of_children_born')
sns.histplot(ax=axes[1, 0], data=df_lr, x='Husband_Occupation')
axes[1,1].axis("off")

axes[0, 0].set(xlabel='Wife_age')
axes[0, 1].set(xlabel='No_of_children_born')
axes[1, 0].set(xlabel='Husband_Occupation')

plt.suptitle('Fig 1: Hist Plots: Wife_age, No_of_children_born, Husband_Occupation'

plt.show()
```



Observations and Insights:

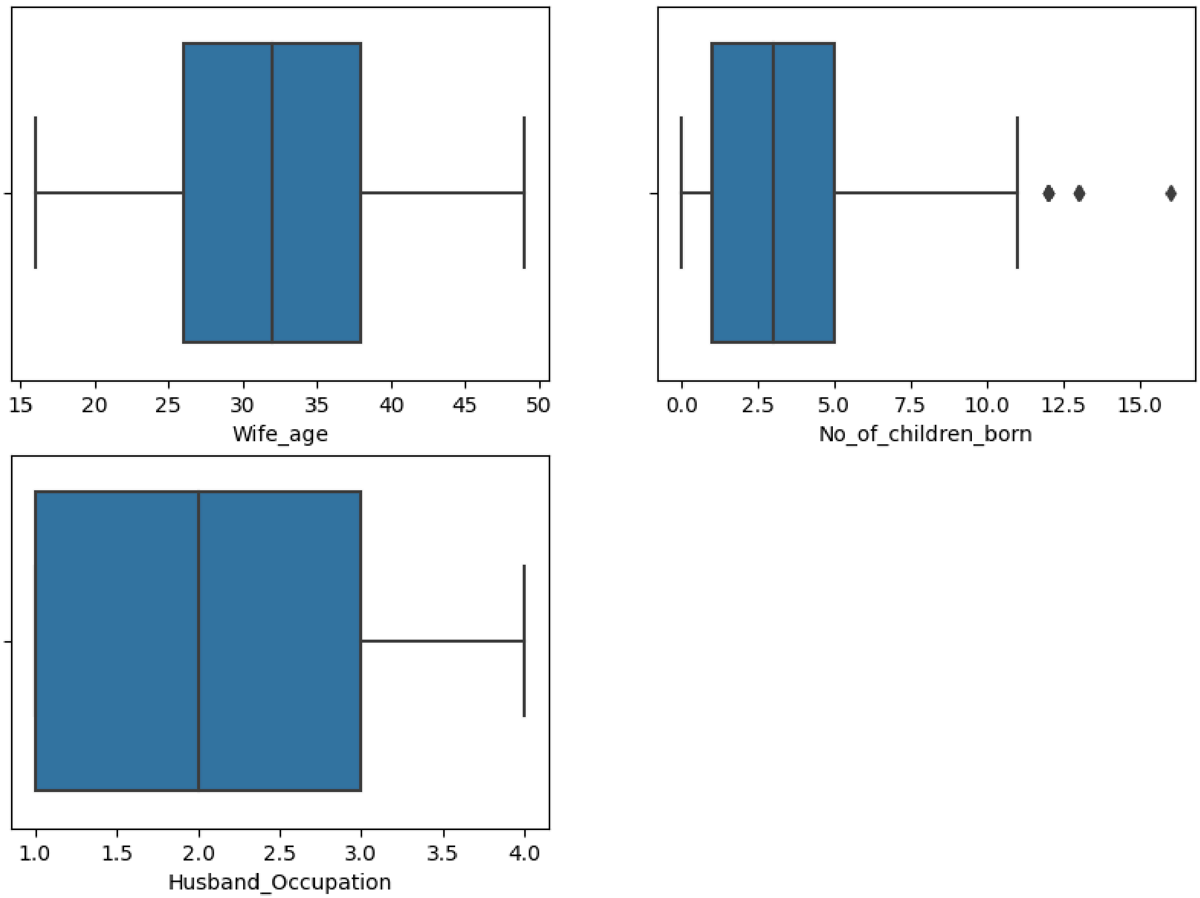
- No distribution (Wife_age, No_of_children_born and Husband_Occupation) is evenly distributed (symmetric).

In [160…

```
# Box Plots for Wife_age, No_of_children_born and Husband_Occupation

fig, axes = plt.subplots(2,2, figsize=(10, 7))

sns.boxplot(ax=axes[0, 0], data=df_lr, x='Wife_age')
```

```
sns.boxplot(ax=axes[0, 1], data=df_lr, x='No_of_children_born')
sns.boxplot(ax=axes[1, 0], data=df_lr, x='Husband_Occupation')
axes[1,1].axis("off")

axes[0, 0].set(xlabel='Wife_age')
axes[0, 1].set(xlabel='No_of_children_born')
axes[1, 0].set(xlabel='Husband_Occupation')

plt.suptitle('Fig 2: Box Plots: Wife_age, No_of_children_born, Husband_Occupation',

plt.show()
```

Fig 2: Box Plots: Wife_age, No_of_children_born, Husband_Occupation



Observations and Insights:

- No_of_children_born column has few outliers.

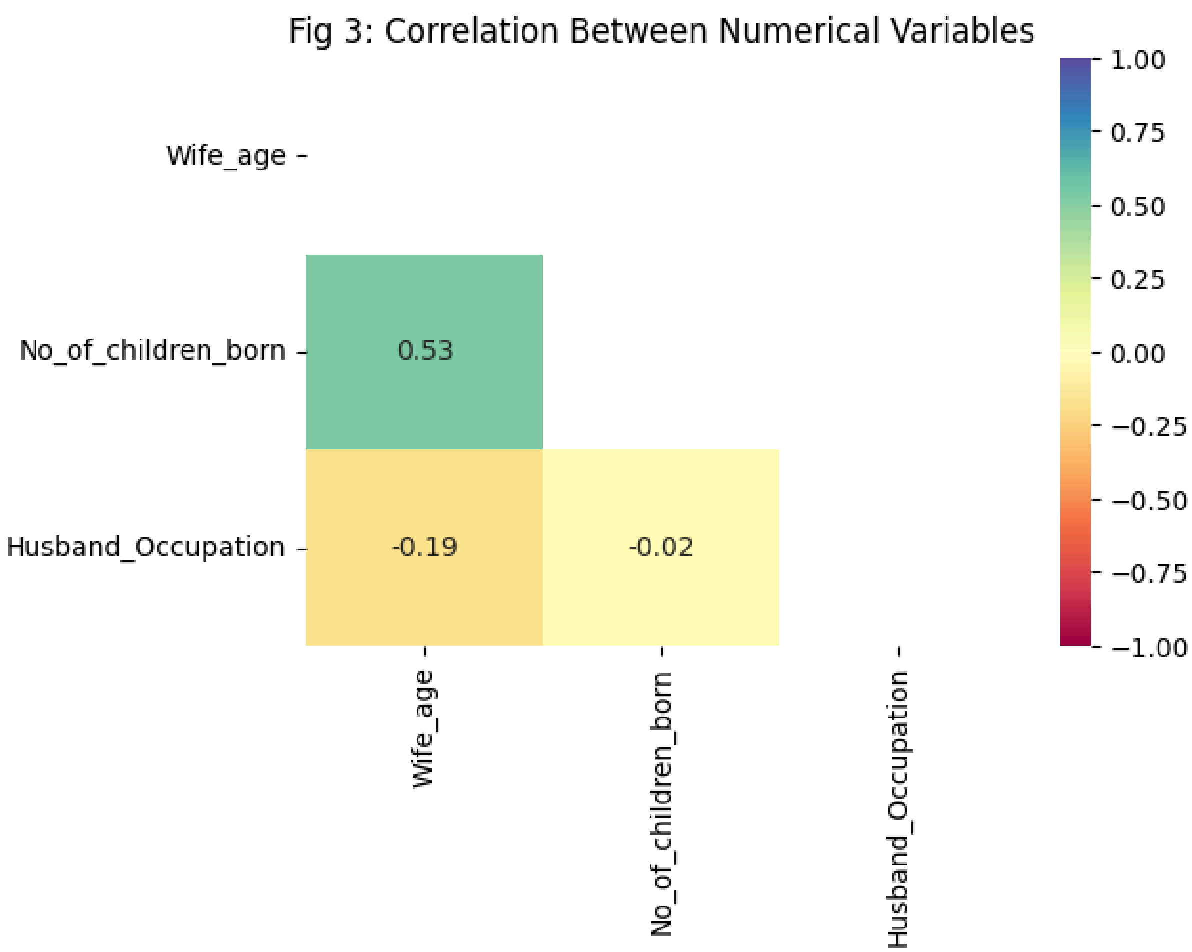
Multivariate Analysis

Correlation Plot

In [161…

```
# Heatmap to plot correlation between all numerical variables in the dataset

df_lr_corr = df_lr.select_dtypes(include=np.number)

corr = df_lr_corr.corr(method='pearson')
```

```
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(6, 4))
sns.heatmap(df_lr_corr.corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectr
plt.title('Fig 3: Correlation Between Numerical Variables')
plt.show()
```

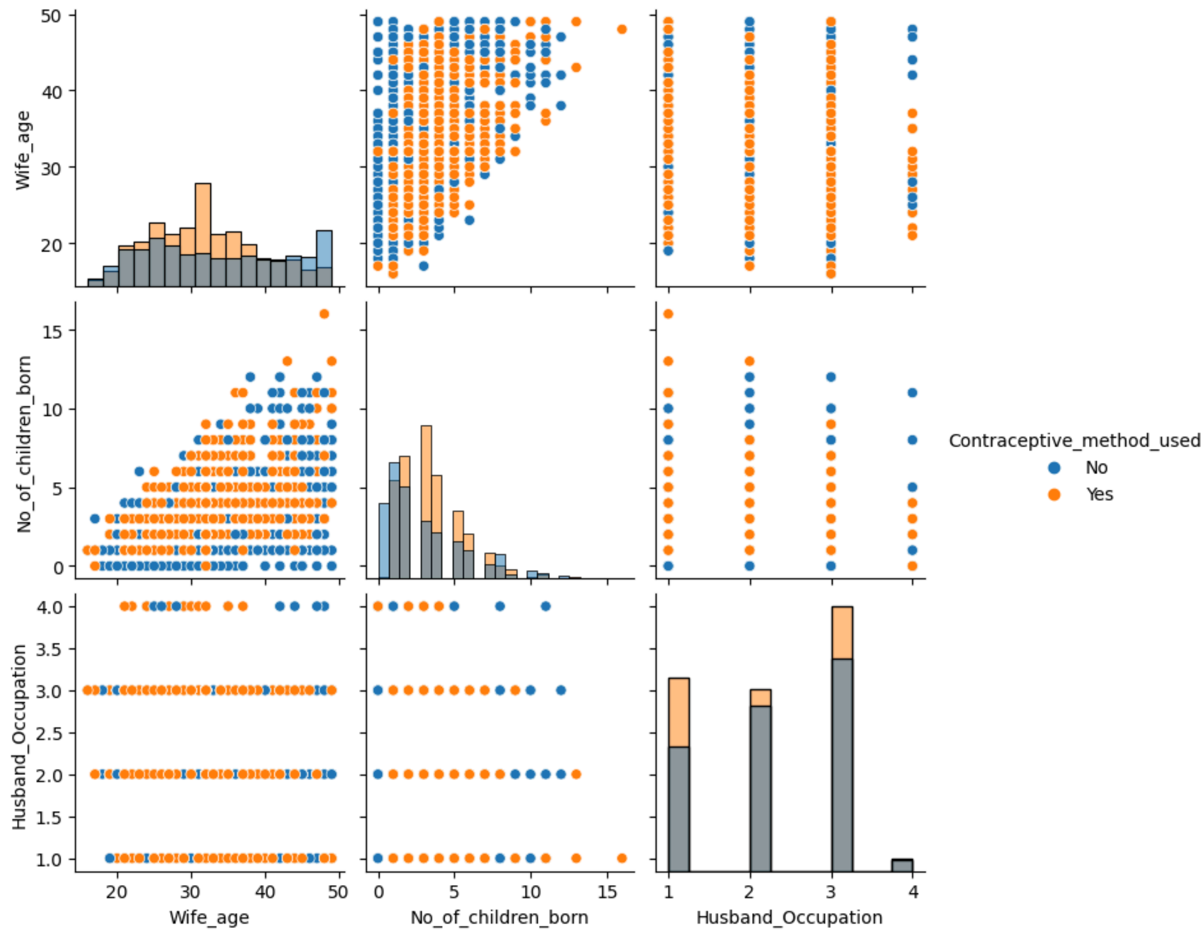


Observations and Insights:

- There is moderate correlation between Wife_age and No_of_children_born.

In [162…

```
# Pair Plot

sns.pairplot(df_lr,diag_kind='hist',hue='Contraceptive_method_used');
```



Converting all objects to categorical codes

In [163…

```
# We are coding up the Wife_education variable in an ordinal manner

df_lr['Wife_education']=np.where(df_lr['Wife_education'] =='Uneducated', '1', df_lr
df_lr['Wife_education']=np.where(df_lr['Wife_education'] =='Primary', '2', df_lr['W
df_lr['Wife_education']=np.where(df_lr['Wife_education'] =='Secondary', '3', df_lr[
df_lr['Wife_education']=np.where(df_lr['Wife_education'] =='Tertiary', '4', df_lr['
```

In [164…

```
# We are coding up the Husband_education variable in an ordinal manner

df_lr['Husband_education']=np.where(df_lr['Husband_education'] =='Uneducated', '1',
df_lr['Husband_education']=np.where(df_lr['Husband_education'] =='Primary', '2', df
df_lr['Husband_education']=np.where(df_lr['Husband_education'] =='Secondary', '3',
df_lr['Husband_education']=np.where(df_lr['Husband_education'] =='Tertiary', '4', d
```

In [165…

```
# We are coding up the Wife_religion variable in an ordinal manner

df_lr['Wife_religion']=np.where(df_lr['Wife_religion'] =='Non-Scientology', '0', df
df_lr['Wife_religion']=np.where(df_lr['Wife_religion'] =='Scientology', '1', df_lr[
```

In [166…

```
# We are coding up the Wife_Working variable in an ordinal manner

df_lr['Wife_Working']=np.where(df_lr['Wife_Working'] =='No', '0', df_lr['Wife_Worki
df_lr['Wife_Working']=np.where(df_lr['Wife_Working'] =='Yes', '1', df_lr['Wife_Work
```

In [167…

```
# We are coding up the Standard_of_living_index variable in an ordinal manner

df_lr['Standard_of_living_index']=np.where(df_lr['Standard_of_living_index'] =='Ver
df_lr['Standard_of_living_index']=np.where(df_lr['Standard_of_living_index'] =='Low
df_lr['Standard_of_living_index']=np.where(df_lr['Standard_of_living_index'] =='Hig
df_lr['Standard_of_living_index']=np.where(df_lr['Standard_of_living_index'] =='Ver
```

In [168…

```
# We are coding up the Media_exposure variable in an ordinal manner

df_lr['Media_exposure']=np.where(df_lr['Media_exposure'] =='Not-Exposed', '0', df_l
df_lr['Media_exposure']=np.where(df_lr['Media_exposure'] =='Exposed', '1', df_lr['M
```

In [169…

```
df_lr.head()
```

Out[169…

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife
0	24.0	2	3	3.0	1	
1	45.0	1	3	10.0	1	
2	43.0	2	3	7.0	1	
3	42.0	3	2	9.0	1	
4	36.0	3	3	8.0	1	

Converting object variables to numeric variables

In [170…

```
## Converting object variables to numeric variables

df_lr['Wife_education'] = df_lr['Wife_education'].astype('int64')
df_lr['Husband_education'] = df_lr['Husband_education'].astype('int64')
df_lr['Wife_religion'] = df_lr['Wife_religion'].astype('int64')
df_lr['Wife_Working'] = df_lr['Wife_Working'].astype('int64')
df_lr['Standard_of_living_index'] = df_lr['Standard_of_living_index'].astype('int64
df_lr['Media_exposure'] = df_lr['Media_exposure'].astype('int64')
```

In [171…

```
df_lr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Wife_age                   1393 non-null   float64
 1   Wife_education             1393 non-null   int64  
 2   Husband_education          1393 non-null   int64  
 3   No_of_children_born        1393 non-null   float64
 4   Wife_religion              1393 non-null   int64  
 5   Wife_Working               1393 non-null   int64  
 6   Husband_Occupation         1393 non-null   int64  
 7   Standard_of_living_index   1393 non-null   int64  
 8   Media_exposure             1393 non-null   int64  
 9   Contraceptive_method_used  1393 non-null   object 
dtypes: float64(2), int64(7), object(1)
memory usage: 119.7+ KB
```

Assigning 0 to Contraceptive_method_used (No) and 1 to Contraceptive_method_used (Yes)

In [172…

```
df_lr['Contraceptive_method_used'].value_counts()
```

Out[172…

```
Contraceptive_method_used
Yes    779
No     614
Name: count, dtype: int64
```

1 is decided to be Contraceptive_method_used (Yes) as that is class of interest as defined by the problem statement

In [173…

```
df_lr['Contraceptive_method_used'] = df_lr['Contraceptive_method_used'].replace({'N
df_lr['Contraceptive_method_used'].value_counts()
```

Out[173…

```
Contraceptive_method_used
1    779
0    614
Name: count, dtype: int64
```

In [174…

```
df_lr.head()
```

Out[174…

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife
0	24.0	2	3	3.0	1	
1	45.0	1	3	10.0	1	
2	43.0	2	3	7.0	1	
3	42.0	3	2	9.0	1	
4	36.0	3	3	8.0	1	

Train-Test Split

In [175…

```
# Copy all the predictor variables into X dataframe
X = df_lr.drop('Contraceptive_method_used', axis=1)

# Copy target into the y dataframe
y = df_lr[['Contraceptive_method_used']]
```

Split X and y into train and test sets in a 70:30 ratio.

In [176…

```
# Split X and y into training and test set in 70:30 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_s
```

In [177…

```
X_train.head()
```

Out[177…

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	W
336	34.0	4	3	0.0	0	
781	37.0	4	4	3.0	1	
433	37.0	4	4	2.0	1	
588	29.0	4	4	2.0	1	
468	24.0	1	4	1.0	1	

In [178…

```
X_test.head()
```

Out[178…

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	W
1012	29.0	3	4	4.0	1	
446	39.0	4	4	3.0	1	
909	31.0	3	3	3.0	1	
1400	32.0	3	4	4.0	1	
486	38.0	4	4	6.0	1	

Logistic Regression Model

In [81]:

```
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
```

Out[81]:

```
▾ LogisticRegression
LogisticRegression()
```

Predicting on Training and Test dataset

In [82]:

```
ytrain_predict = lr_model.predict(X_train)
```

```
ytest_predict = lr_model.predict(X_test)
```

Getting the Predicted Classes and Prob

In [83]:
```
ytest_predict_prob=lr_model.predict_proba(X_test)
pd.DataFrame(ytest_predict_prob).head()
```

Out[83]:

	0	1
0	0.270919	0.729081
1	0.443701	0.556299
2	0.609765	0.390235
3	0.260860	0.739140
4	0.193044	0.806956

Model Evaluation - Training Data

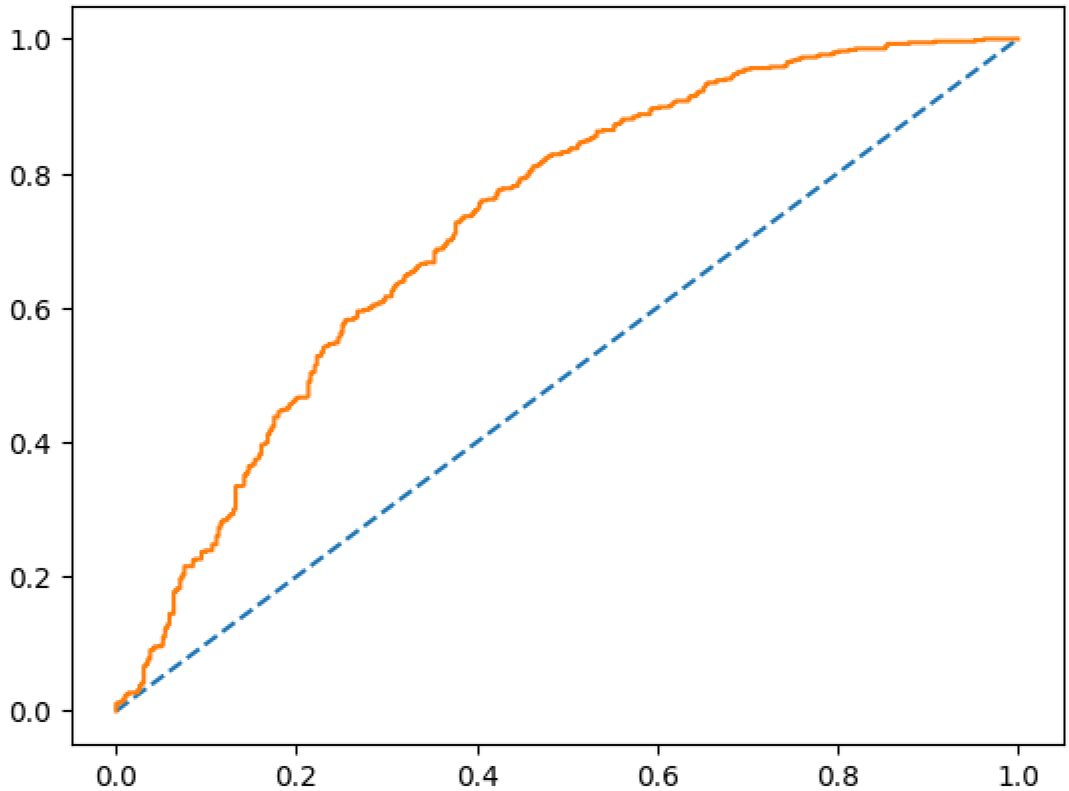
In [84]:
```
# Accuracy - Training Data
lr_model.score(X_train, y_train)
```

Out[84]: 0.6912820512820513

AUC and ROC for the training data

In [85]:
```
# predict probabilities
probs = lr_model.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y_train, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
train_fpr, train_tpr, train_thresholds = roc_curve(y_train, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(train_fpr, train_tpr);
```

```
AUC: 0.725
```



Model Evaluation - Test Data

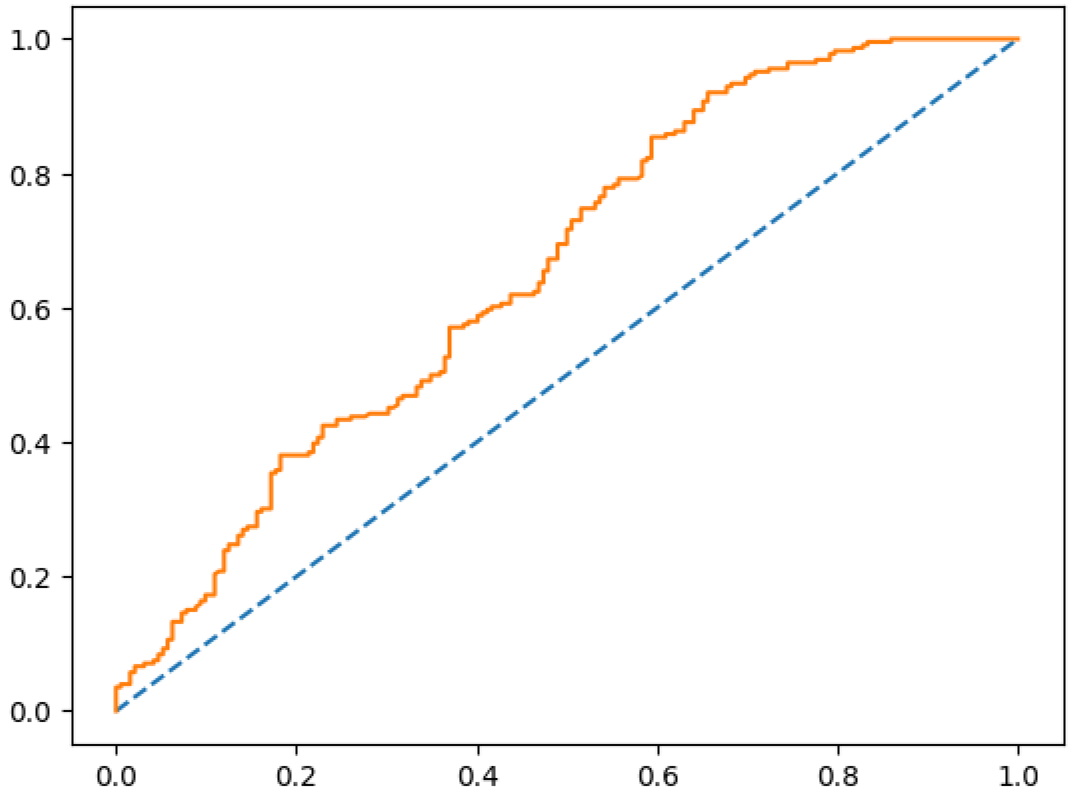
In [86]:
```
# Accuracy - Test Data
lr_model.score(X_test, y_test)
```

Out[86]: 0.6291866028708134

AUC and ROC for the test data

In [87]:
```
# predict probabilities
probs = lr_model.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
test_auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr);
```

AUC: 0.725



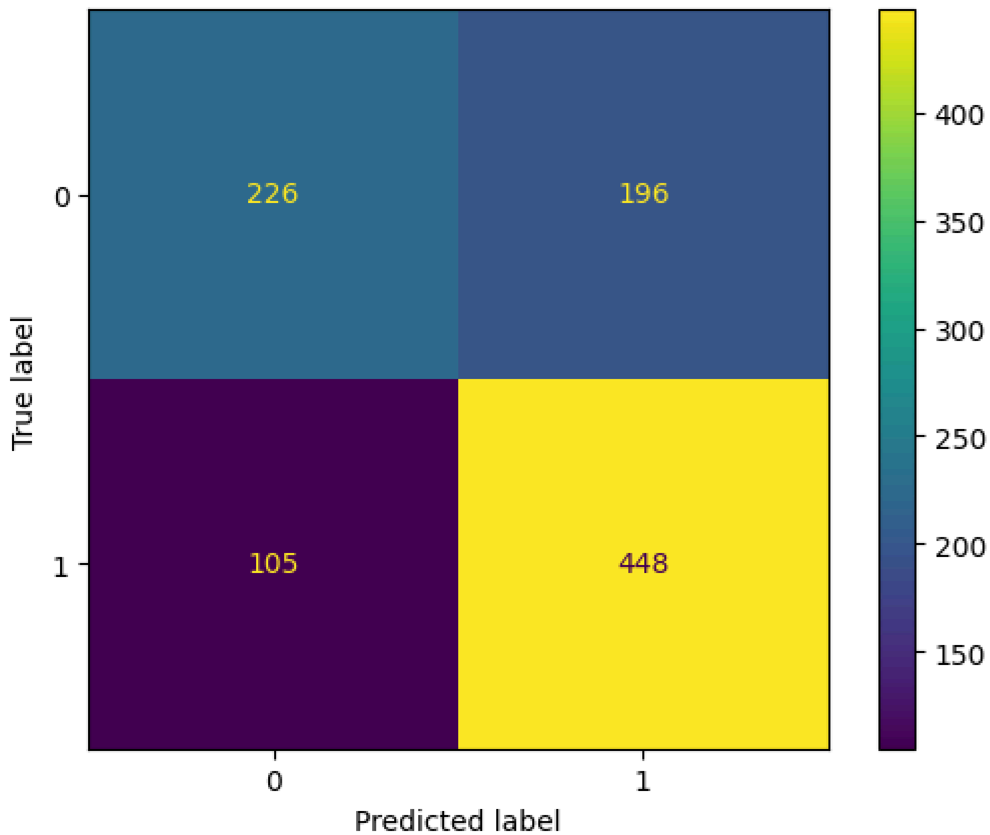
Confusion Matrix for the training data

In [88]:
```
cm_train = confusion_matrix(y_train, ytrain_predict)
cm_train
```

Out[88]:
```
array([[226, 196],
       [105, 448]], dtype=int64)
```

In [89]:
```
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train, display_labels=lr_model.cl
disp.plot()
```

Out[89]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b85370b610>
```



In [90]:
```
print(classification_report(y_train, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.68      0.54      0.60       422
           1       0.70      0.81      0.75       553

    accuracy                           0.69       975
   macro avg       0.69      0.67      0.67       975
weighted avg       0.69      0.69      0.68       975
```

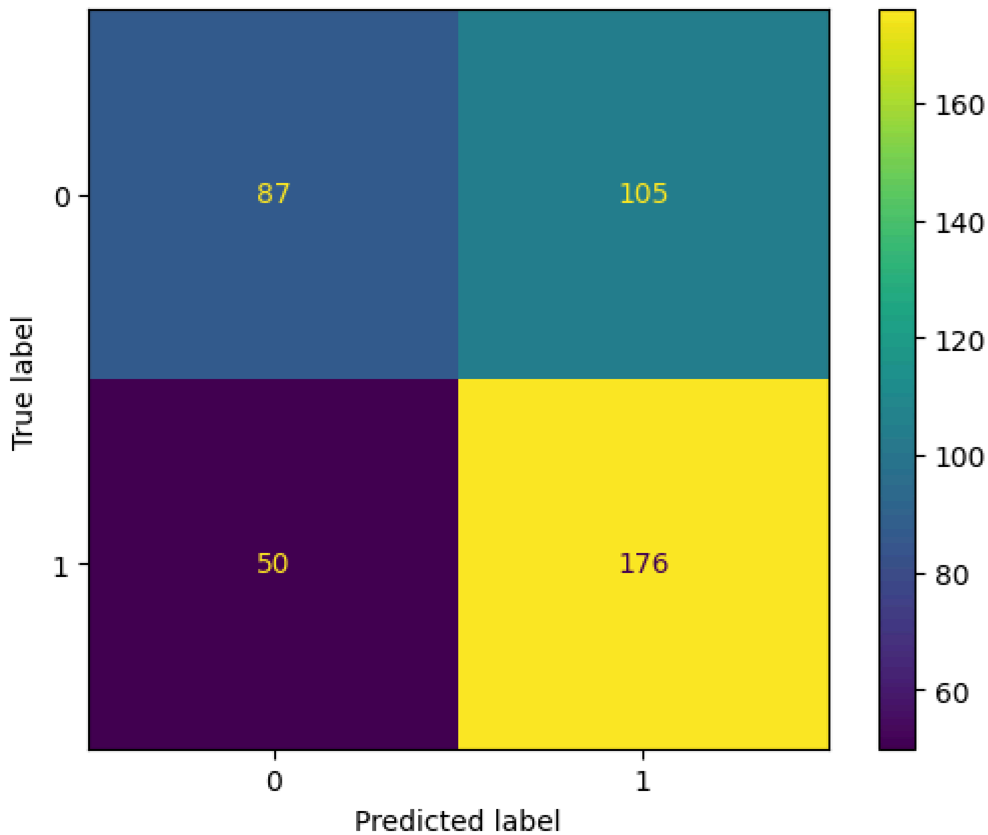
Confusion Matrix for the test data

In [91]:
```
cm_test = confusion_matrix(y_test, ytest_predict)
cm_test
```

Out[91]:
```
array([[ 87, 105],
       [ 50, 176]], dtype=int64)
```

In [92]:
```
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=lr_model.cla
disp.plot()
```

Out[92]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b85370a1a0>



In [93]:
```
print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.64      0.45      0.53       192
           1       0.63      0.78      0.69       226

    accuracy                           0.63       418
   macro avg       0.63      0.62      0.61       418
weighted avg       0.63      0.63      0.62       418
```

Inferences:

For predicting contraceptive method used (Label 1):

Precision (63%) – 63% of wife's predicted are actually using contraceptive method out of all wife's predicted to use contraceptive method.

Recall (78%) – Out of all the wife's actually using contraceptive method, 78% of wife's have been predicted to use contraceptive method.

For predicting contraceptive method not used (Label 0):

Precision (64%) – 64% of wife's predicted are actually not using contraceptive method out of all wife's predicted not using contraceptive method.

Recall (45%) – Out of all the wife's actually not using contraceptive method, 45% of wife's have been predicted not using contraceptive method.

Overall accuracy of the model – 63% of total predictions are correct

Linear Discriminant Analysis Model

In [98]:
```
#Build LDA Model

clf = LinearDiscriminantAnalysis()
lda_model=clf.fit(X_train, y_train)
lda_model
```

Out[98]:
```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

Predicting on Training and Test dataset

In [99]:
```
ytrain_predict = lda_model.predict(X_train)
ytest_predict = lda_model.predict(X_test)
```

Getting the Predicted Classes and Prob

In [100…
```
ytest_predict_prob=lda_model.predict_proba(X_test)
pd.DataFrame(ytest_predict_prob).head()
```

Out[100…

	0	1
0	0.273441	0.726559
1	0.431870	0.568130
2	0.587760	0.412240
3	0.255022	0.744978
4	0.190598	0.809402

Model Evaluation - Training Data

In [101…
```
# Accuracy - Training Data
lda_model.score(X_train, y_train)
```

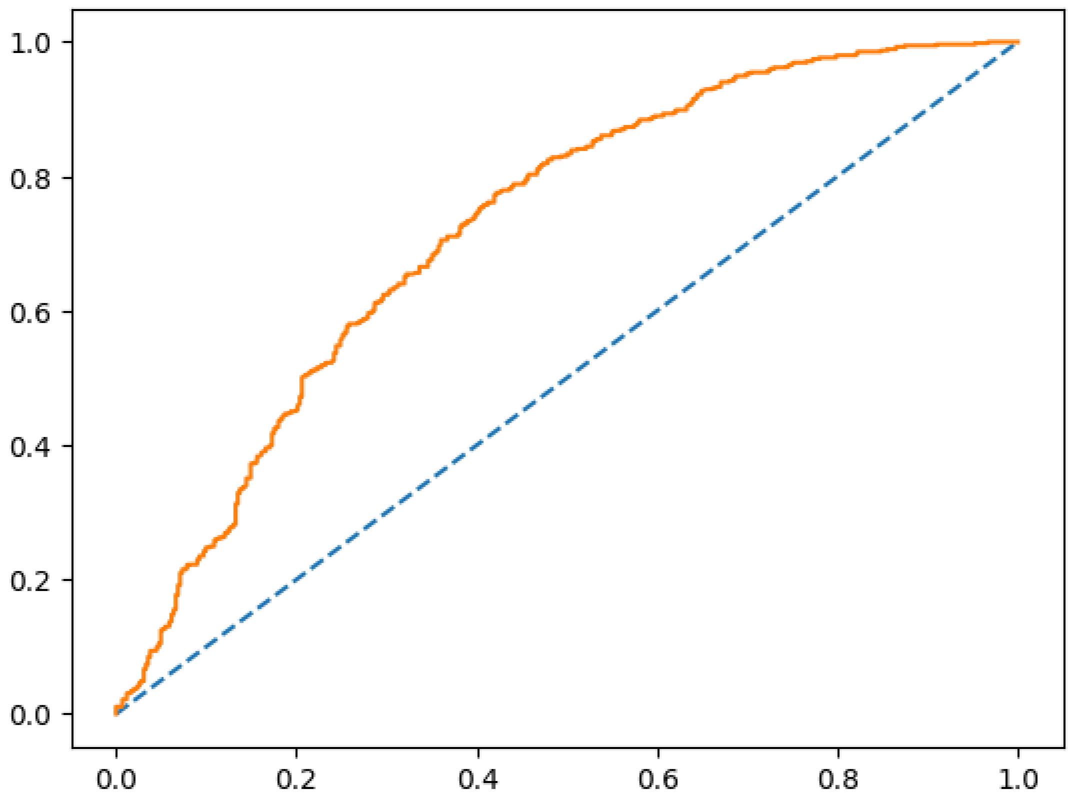
Out[101…
```
0.6923076923076923
```

AUC and ROC for the training data

In [102…
```
# predict probabilities
probs = lda_model.predict_proba(X_train)
```

```python
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(y_train, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
train_fpr, train_tpr, train_thresholds = roc_curve(y_train, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(train_fpr, train_tpr);
```

```
AUC: 0.724
```



Model Evaluation - Test Data

In [103…

```python
# Accuracy - Test Data
lda_model.score(X_test, y_test)
```

Out[103…

```
0.6220095693779905
```

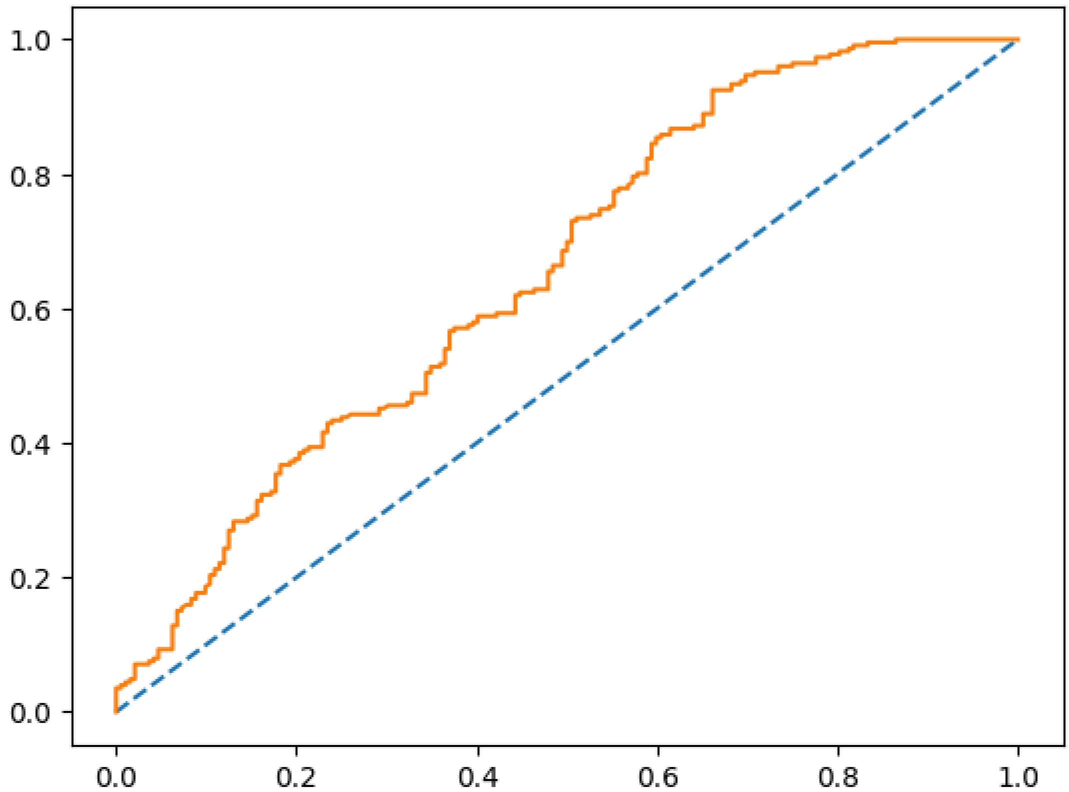
AUC and ROC for the test data

In [104…

```python
# predict probabilities
probs = lda_model.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
test_auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
```

```python
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr);
```

AUC: 0.724



Confusion Matrix for the training data

In [105…

```python
cm_train = confusion_matrix(y_train, ytrain_predict)
cm_train
```

Out[105…

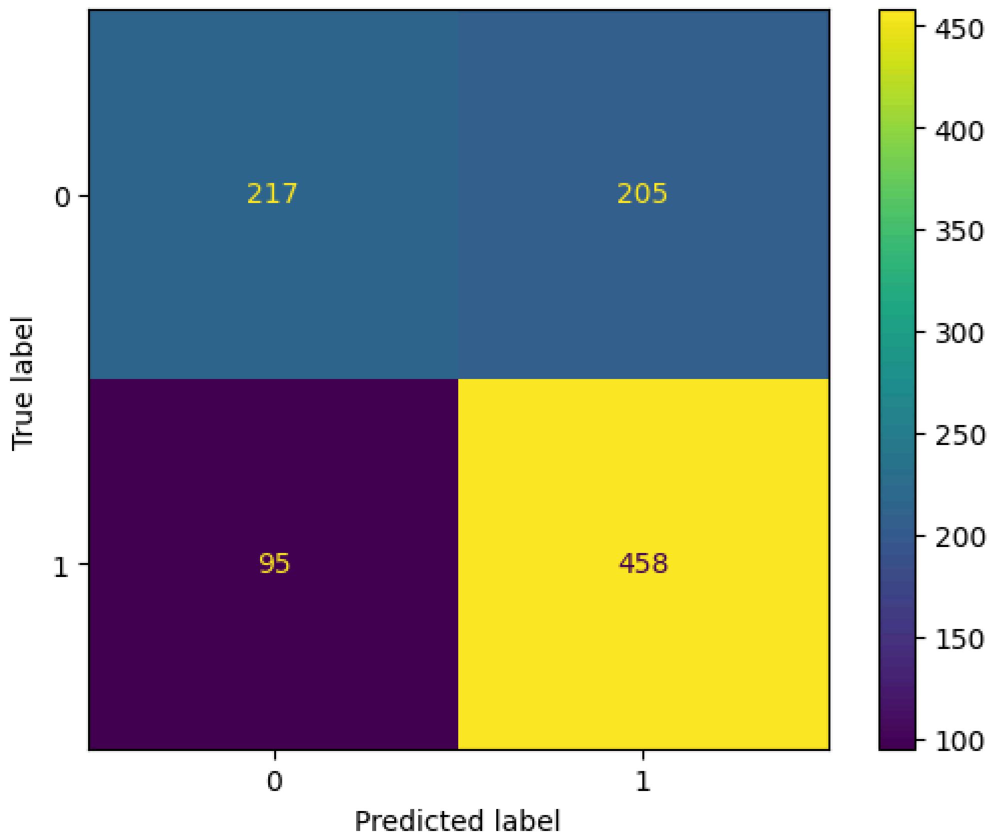
```
array([[217, 205],
       [ 95, 458]], dtype=int64)
```

In [106…

```python
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train, display_labels=lda_model.c
disp.plot()
```

Out[106…

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b856429000>
```



In [107…

```
print(classification_report(y_train, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.70      0.51      0.59       422
           1       0.69      0.83      0.75       553

    accuracy                           0.69       975
   macro avg       0.69      0.67      0.67       975
weighted avg       0.69      0.69      0.68       975
```

Confusion Matrix for the test data

In [108…

```
cm_test = confusion_matrix(y_test, ytest_predict)
cm_test
```

Out[108…

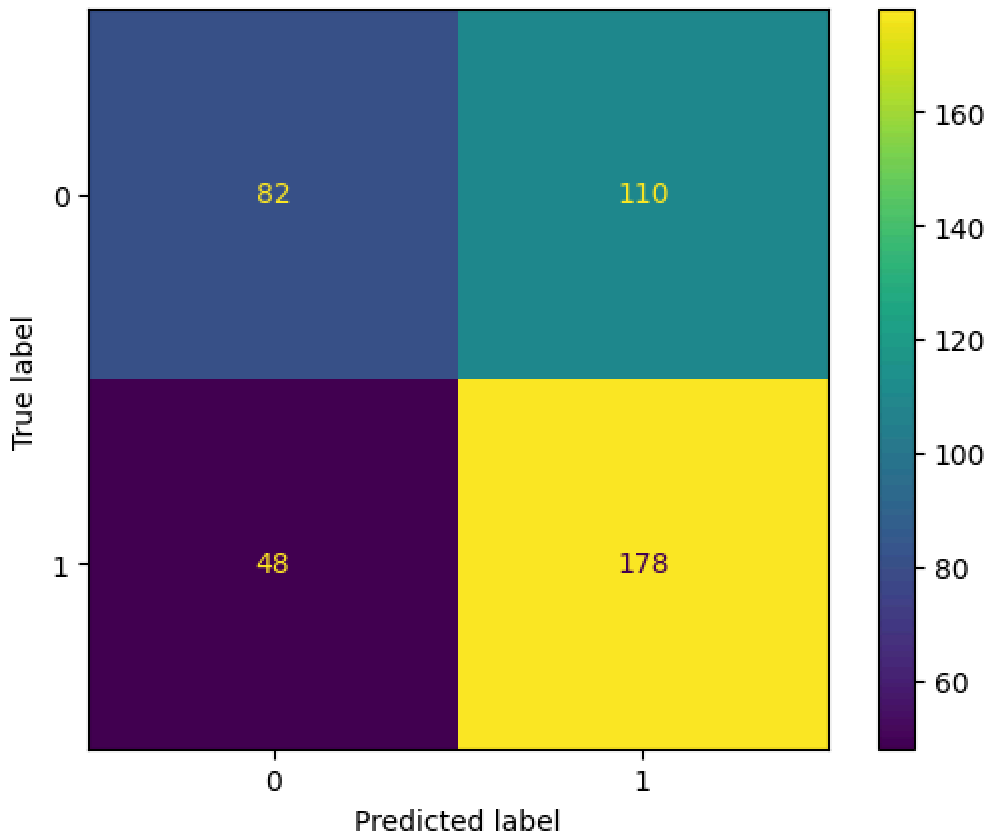
```
array([[ 82, 110],
       [ 48, 178]], dtype=int64)
```

In [109…

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=lda_model.cl
disp.plot()
```

Out[109…

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b8563d2e60>
```



In [110…

```
print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.63      0.43      0.51       192
           1       0.62      0.79      0.69       226

    accuracy                           0.62       418
   macro avg       0.62      0.61      0.60       418
weighted avg       0.62      0.62      0.61       418
```

Inferences:

For predicting contraceptive method used (Label 1):

Precision (62%) – 62% of wife's predicted are actually using contraceptive method out of all wife's predicted to use contraceptive method.

Recall (79%) – Out of all the wife's actually using contraceptive method, 79% of wife's have been predicted to use contraceptive method.

For predicting contraceptive method not used (Label 0):

Precision (63%) – 63% of wife's predicted are actually not using contraceptive method out of all wife's predicted not using contraceptive method.

Recall (43%) – Out of all the wife's actually not using contraceptive method, 43% of wife's have been predicted not using contraceptive method.

Overall accuracy of the model – 62% of total predictions are correct

Building a Decision Tree Classifier (CART)

In [180…
```
# Initialise a Decision Tree Classifier
dt_model = DecisionTreeClassifier(criterion = 'gini', random_state=1)
```

In [181…
```
dt_model.fit(X_train, y_train)
dt_model
```

Out[181…
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=1)
```

In [182…
```
train_char_label = ['No', 'Yes']

cmu_tree = open('d:\cmu_tree.dot','w')
dot_data = tree.export_graphviz(dt_model, out_file=cmu_tree, feature_names = list(X

cmu_tree.close()
```

Variable Importance

In [183…
```
# importance of features in the tree building

print (pd.DataFrame(dt_model.feature_importances_, columns = ["Imp"], index = X_tra
```

```
                                Imp
Wife_age                   0.292197
No_of_children_born        0.240709
Wife_education             0.106970
Husband_Occupation         0.106614
Standard_of_living_index   0.103483
Husband_education          0.051409
Wife_Working               0.048122
Wife_religion              0.033211
Media_exposure             0.017285
```

Predicting Test Data

In [184…
```
y_predict = dt_model.predict(X_test)
```

In [185…
```
y_predict.shape
```

Out[185…
```
(418,)
```

Regularising the Decision Tree

```
In [186…
param_grid = {'max_features': ['auto', 'sqrt', 'log2'],
              'ccp_alpha': [0.1, .01, .001],
              'max_depth' : [1,5,10,15,20],
              'min_samples_leaf':[1,5,10,15,20],
              'criterion' :['gini', 'entropy']
             }
tree_clas = DecisionTreeClassifier(random_state=1)
grid_search = GridSearchCV(estimator=tree_clas, param_grid=param_grid, cv=5, verbos
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 450 candidates, totalling 2250 fits
```

```
Out[186…
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
In [187…
grid_search.best_estimator_
```

```
Out[187…
DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.001, max_depth=10, max_features='sqrt',
                       min_samples_leaf=15, random_state=1)
```

```
In [188…
reg_dt_model = DecisionTreeClassifier(ccp_alpha=0.001, max_depth=10,max_features='s
                                      min_samples_leaf=15, random_state=1)

reg_dt_model.fit(X_train, y_train)
```

```
Out[188…
DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.001, max_depth=10, max_features='sqrt',
                       min_samples_leaf=15, random_state=1)
```

Generating New Tree

```
In [189…
train_char_label = ['No', 'Yes']

cmu_tree_regularized = open('d:\cmu_tree_regularized.dot','w')
dot_data = tree.export_graphviz(reg_dt_model, out_file= cmu_tree_regularized, featu

cmu_tree_regularized.close()
```

Variable Importance

```
In [190…
# importance of features in the tree building

print (pd.DataFrame(reg_dt_model.feature_importances_, columns = ["Imp"], index = X
```

```
                              Imp
No_of_children_born       0.459324
Wife_age                  0.317840
Wife_education            0.155576
Husband_education         0.027978
Standard_of_living_index  0.017100
Wife_Working              0.011926
Husband_Occupation        0.010256
Wife_religion             0.000000
Media_exposure            0.000000
```

Predicting on Training and Test dataset

In [191...

```
ytrain_predict = reg_dt_model.predict(X_train)
ytest_predict = reg_dt_model.predict(X_test)
```

In [192...

```
print('ytrain_predict',ytrain_predict.shape)
print('ytest_predict',ytest_predict.shape)
```

```
ytrain_predict (975,)
ytest_predict (418,)
```

Getting the Predicted Probabilities

In [193...

```
ytest_predict_prob=reg_dt_model.predict_proba(X_test)
```

In [194...

```
pd.DataFrame(ytest_predict_prob).head()
```

Out[194...

	0	1
0	0.172840	0.827160
1	0.147887	0.852113
2	0.307692	0.692308
3	0.172840	0.827160
4	0.147887	0.852113

Model Evaluation - Training Data

In [195...

```
# Accuracy - Training Data
reg_dt_model.score(X_train, y_train)
```

Out[195...

```
0.7446153846153846
```

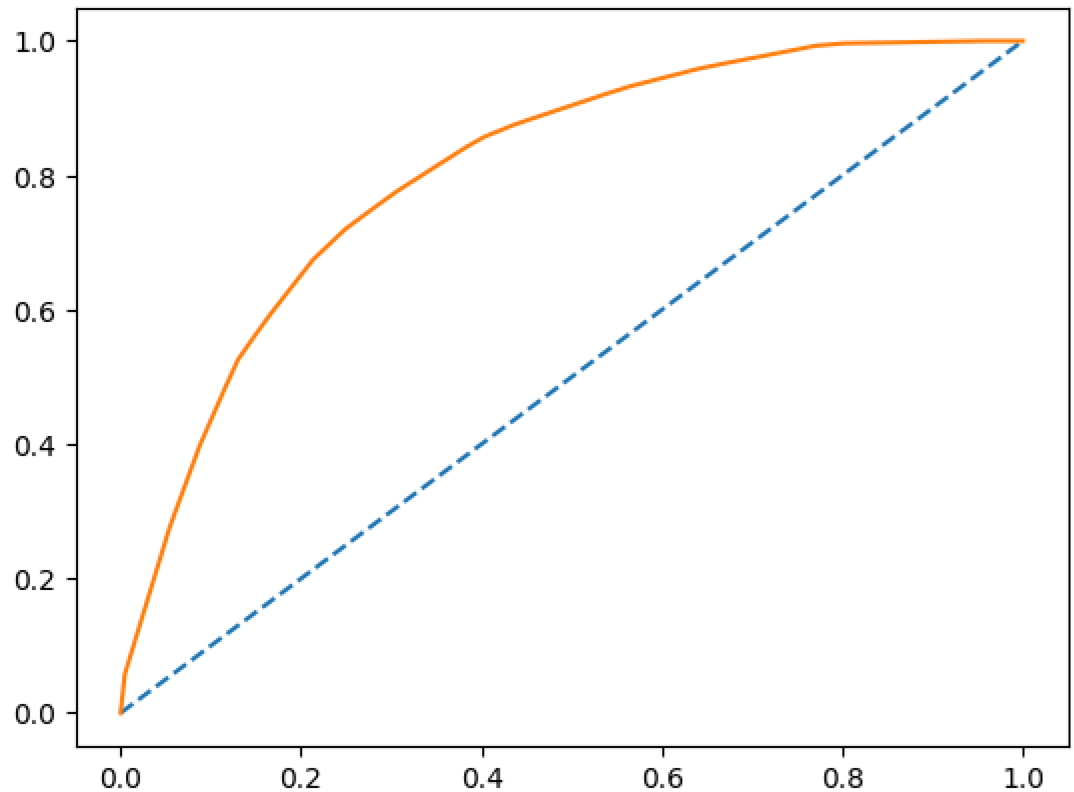
AUC and ROC for the training data

In [196...

```
# predict probabilities
probs = reg_dt_model.predict_proba(X_train)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
```

```
# calculate AUC
auc = roc_auc_score(y_train, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
train_fpr, train_tpr, train_thresholds = roc_curve(y_train, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(train_fpr, train_tpr);
```

```
AUC: 0.809
```



Model Evaluation - Test Data

In [197…

```
# Accuracy - Test Data
reg_dt_model.score(X_test, y_test)
```

Out[197…

```
0.6746411483253588
```

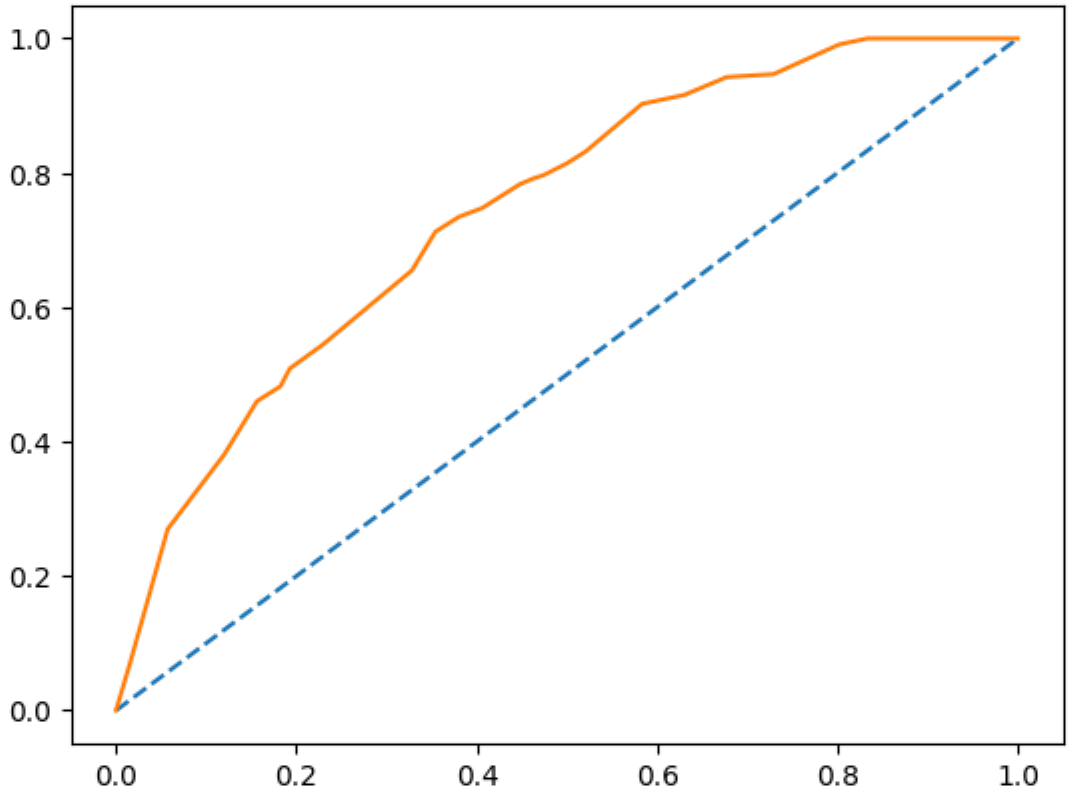
AUC and ROC for the test data

In [198…

```
# predict probabilities
probs = reg_dt_model.predict_proba(X_test)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
test_auc = roc_auc_score(y_test, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, probs)
plt.plot([0, 1], [0, 1], linestyle='--')
```

```python
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr);
```

```
AUC: 0.809
```



Confusion Matrix for the training data

In [199…

```python
cm_train = confusion_matrix(y_train, ytrain_predict)
cm_train
```

Out[199…

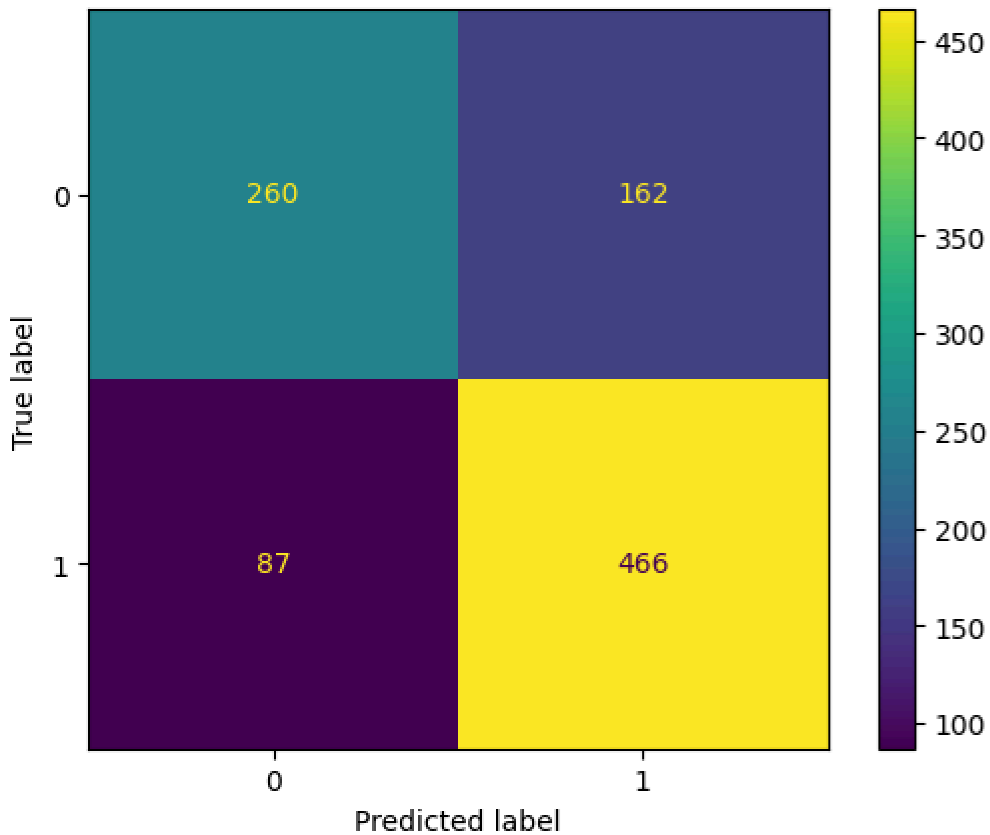
```
array([[260, 162],
       [ 87, 466]], dtype=int64)
```

In [200…

```python
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train, display_labels=reg_dt_mode
disp.plot()
```

Out[200…

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b85303dc60>
```



In [201…

```
print(classification_report(y_train, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.75      0.62      0.68       422
           1       0.74      0.84      0.79       553

    accuracy                           0.74       975
   macro avg       0.75      0.73      0.73       975
weighted avg       0.75      0.74      0.74       975
```

Confusion Matrix for the test data

In [202…

```
cm_test = confusion_matrix(y_test, ytest_predict)
cm_test
```

Out[202…

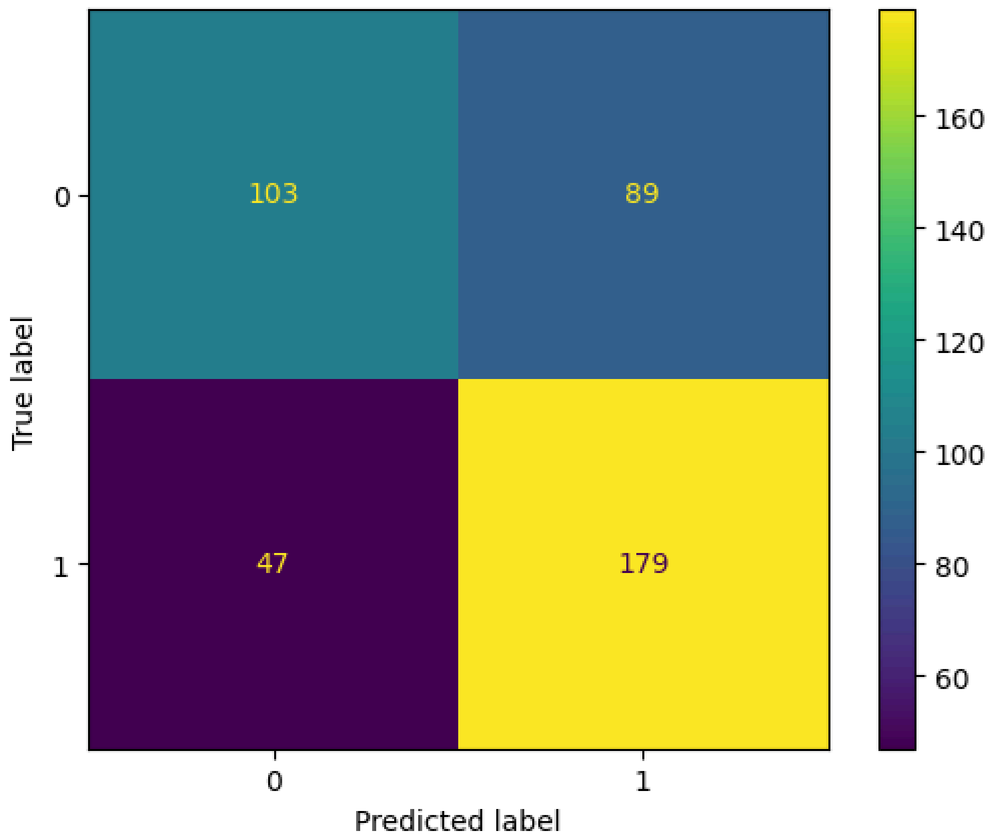
```
array([[103,  89],
       [ 47, 179]], dtype=int64)
```

In [203…

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=lda_model.cl
disp.plot()
```

Out[203…

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b85870cbe0>
```



In [204…

```
print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.69      0.54      0.60       192
           1       0.67      0.79      0.72       226

    accuracy                           0.67       418
   macro avg       0.68      0.66      0.66       418
weighted avg       0.68      0.67      0.67       418
```

Inferences:

For predicting contraceptive method used (Label 1):

Precision (67%) – 67% of wife's predicted are actually using contraceptive method out of all wife's predicted to use contraceptive method.

Recall (79%) – Out of all the wife's actually using contraceptive method, 79% of wife's have been predicted to use contraceptive method.

For predicting contraceptive method not used (Label 0):

Precision (69%) – 69% of wife's predicted are actually not using contraceptive method out of all wife's predicted not using contraceptive method.

Recall (54%) – Out of all the wife's actually not using contraceptive method, 54% of wife's have been predicted not using contraceptive method.

Overall accuracy of the model – 67% of total predictions are correct

Conclusion

Best Model: **Decision Tree Classifier (CART)**

Rationale:

- Accuracy on test data (Logistic Regression Model: 63%, Linear Discriminant Analysis Model: 62%, Decision Tree Classifier (CART): 67%)
- Decision Tree Classifier (CART) Model: Accuracy, AUC, Precision and Recall for test data is almost inline with training data. This proves no overfitting or underfitting has happened, and overall the model is a good model for classification.

No_of_children_born, Wife_age, Husband_education, Wife_education, Standard_of_living_index, Wife_Working and Husband_Occupation (in same order of preference) are the most important variables in determining if a wife will use contraceptive method.

Actionable Insights and Recommendations:

- If a house has more children than there is high probability that wife will use contraceptive method to avoid pregnancy.
- Use of contraceptive method is higher for the wife whose husband is highly educated.
- Use of contraceptive method is higher for the wife whose age is also higher.
- Use of contraceptive method is higher for the wife who is highly educated.
- Use of contraceptive method is higher for the wife whose standard of living index is high.
- Use of contraceptive method is higher for the wife who is working.

In [ ]:
```

```