DA_Fall21_HW_3 Support Vector Machine and Decision Trees

Due on 11/22 23:59 pm

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

We will use the same affair dataset from HW2, but will skip the EDA phrase we have done enough of it

Everything removing outliers, create dummies variabes had been done for you

In [5]:

```
# Remember the affair data set from HW3, we will use that dataset again
# but we will directly load it from the API
orig_df = pd.read_csv("affairs2.csv")
# Set up our target class label
orig_df['had_affair'] = orig_df['affairs'].apply(lambda x: 1 if x != 0 else 0)
orig_df = orig_df.drop('affairs',axis=1)
# remove NA
orig_df.dropna(inplace=True)
# make sure there is no missing values
orig_df.isnull().sum()
```

Out[5]:

```
rate_marriage      0
age                0
yrs_married        0
children           0
religious          0
educ               0
occupation         0
occupation_husb    0
had_affair         0
dtype: int64
```

In [6]:

```python
# separate the features into categorical vs numerical
numerical_features = ['age', 'yrs_married', 'children']
categorical_features = ['rate_marriage', 'religious', 'educ', 'occupation', 'occupation_husb
# collect all numerical features with the target variables first
numerical_df = orig_df[numerical_features + ['had_affair']]
numerical_df.head()
```

Out[6]:

	age	yrs_married	children	had_affair
0	32.0	9.0	3.0	1
1	27.0	13.0	3.0	1
2	22.0	2.5	0.0	1
3	37.0	16.5	4.0	1
4	27.0	9.0	1.0	1

In [7]:

```python
# create corresponding dummies variables
rate_marriage = pd.get_dummies(orig_df['rate_marriage'],drop_first=True)
religious = pd.get_dummies(orig_df['religious'],drop_first=True)
edu = pd.get_dummies(orig_df['educ'],drop_first=True)
occ = pd.get_dummies(orig_df['occupation'],drop_first=True)
husb_occ = pd.get_dummies(orig_df['occupation_husb'],drop_first=True)
```

In [8]:

```python
rate_marriage.head()
```

Out[8]:

	2.0	3.0	4.0	5.0
0	0	1	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	1	0
4	0	0	0	1

In [9]:

```
religious.head()
```

Out[9]:

	2.0	3.0	4.0
0	0	1	0
1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	0

In [10]:

```
# better to create a header to avoid same name
rate_marriage.columns = ['rate1','rate2','rate3','rate4']
rate_marriage
```

Out[10]:

	rate1	rate2	rate3	rate4
0	0	1	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	1	0
4	0	0	0	1
...	...	...	...	...
6466	0	0	0	1
6467	0	0	1	0
6468	0	0	0	1
6469	0	0	0	1
6470	0	0	1	0

6366 rows × 4 columns

In [11]:

```
religious.columns = ['re1','rel2','rel3']
religious.head()
```

Out[11]:

	re1	rel2	rel3
0	0	1	0
1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	0

Now we can concatnate the numerical features with rate_marriage and religious variabes

In [12]:

```
df = pd.concat([numerical_df, rate_marriage, religious], axis=1)
df.head()
```

Out[12]:

	age	yrs_married	children	had_affair	rate1	rate2	rate3	rate4	re1	rel2	rel3
0	32.0	9.0	3.0	1	0	1	0	0	0	1	0
1	27.0	13.0	3.0	1	0	1	0	0	0	0	0
2	22.0	2.5	0.0	1	0	0	1	0	0	0	0
3	37.0	16.5	4.0	1	0	0	1	0	0	1	0
4	27.0	9.0	1.0	1	0	0	0	1	0	0	0

The goal of this homework is to practice building Support Vector Machine and Decision Tree Models.

Part A: Support Vector Machine

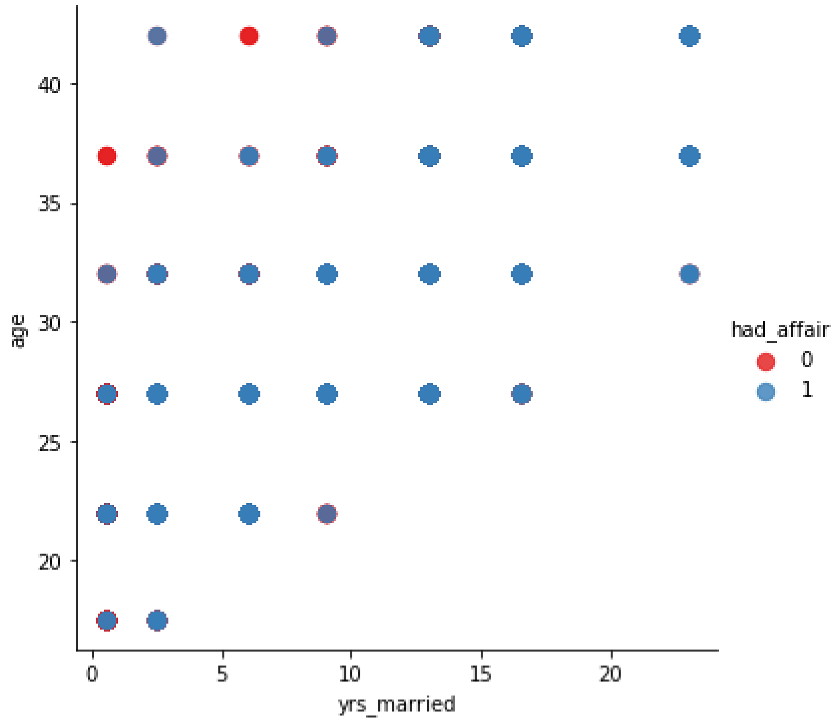
Follow the standard way of building a model and in particular,

1. Build a classification model using SVC using Linear Kernel without specifying the c-parameter using the above provided data frame
2. Try different values of C-parameters (at least one small and one bigger value)
3. Try using rbf as your kernel and use Gamma of 2**-5, 0.1, 1 and 2 with default value for C-parameter
4. Answer the question out of all the models above, what is the best choice for the kernel, C and gamma parameters Explain briefly the effect of using different parameter values

Type your answers and code here

In [13]:

```
sns.lmplot('yrs_married', 'age', data=df, hue='had_affair', palette='Set1', fit_reg=False,
```



In [14]:

```
X = df[['yrs_married', 'age']].values
X
```

Out[14]:

```
array([[ 9. , 32. ],
       [13. , 27. ],
       [ 2.5, 22. ],
       ...,
       [ 2.5, 22. ],
       [ 6. , 32. ],
       [ 2.5, 22. ]])
```

In [15]:

```
Y = df['had_affair'].values
Y
```

Out[15]:

```
array([1, 1, 1, ..., 0, 0, 0], dtype=int64)
```

In [16]:

```
from sklearn import svm
```

In [17]:

```
model = svm.SVC(kernel='linear')
model.fit(X, Y)
```

Out[17]:

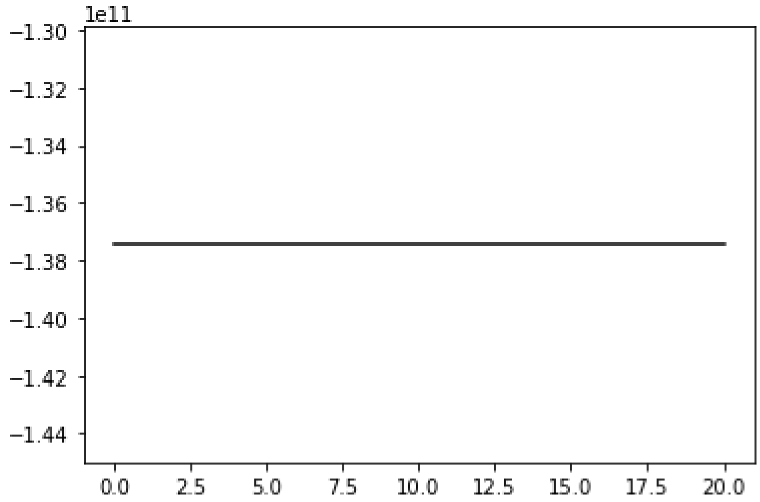
SVC(kernel='linear')

In [18]:

```
w = model.coef_[0]
a = - (w[0] / w[1])
xx = np.linspace(0, 20)
yy = a * xx - ((model.intercept_[0]) / w[1])

```

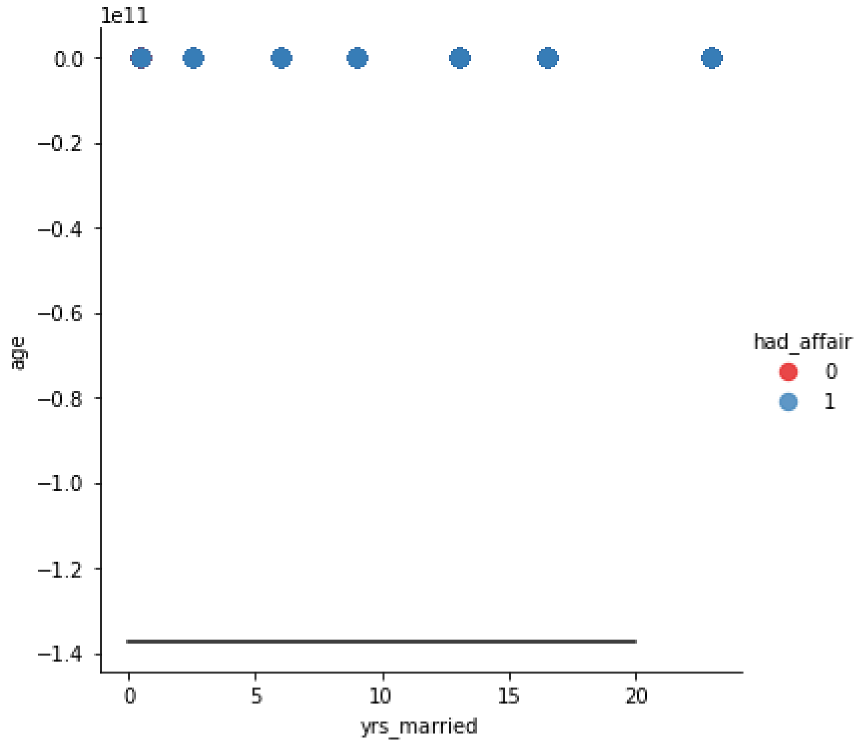
In [19]:

```
plt.plot(xx, yy, color='black');
```



In [20]:

```
sns.lmplot('yrs_married', 'age', data=df, hue='had_affair', palette='Set1', fit_reg=False,
plt.plot(xx, yy, color='black');
```



2

In [21]:

```
model2 = svm.SVC(kernel='linear', C=2**-10)
model2.fit(X, Y)
```

Out[21]:

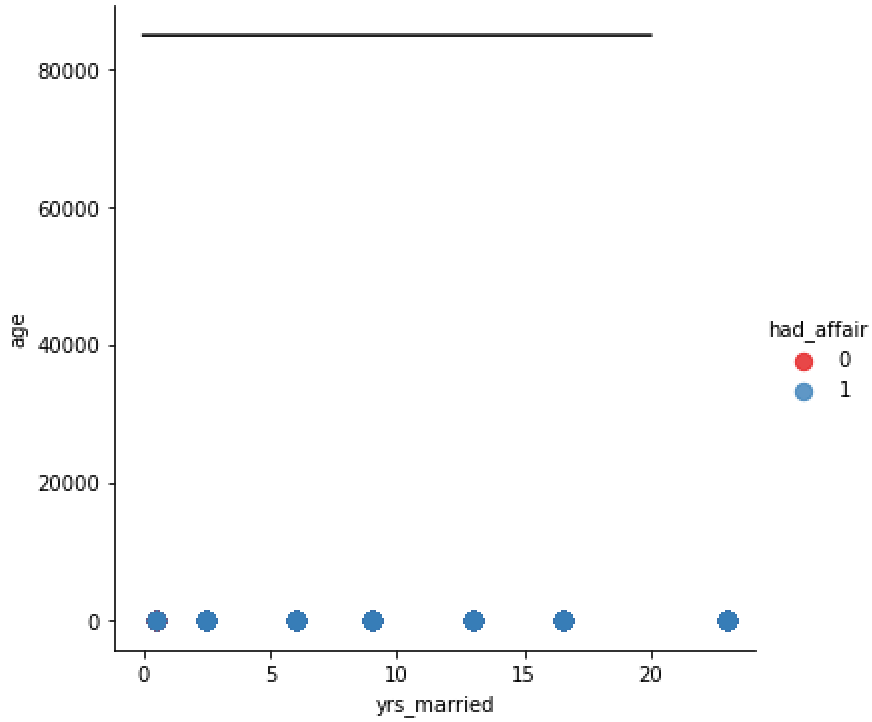
```
SVC(C=0.0009765625, kernel='linear')
```

In [22]:

```
w = model2.coef_[0]
a = - (w[0] / w[1])
xx = np.linspace(0, 20)
yy = a * xx - ((model2.intercept_[0]) / w[1])
```

In [23]:

```
sns.lmplot('yrs_married', 'age', data=df, hue='had_affair', palette='Set1', fit_reg=False,
plt.plot(xx, yy, color='black');
```



In [24]:

```
model3 = svm.SVC(kernel='linear', C=2**3)
model3.fit(X, Y)
```

Out[24]:

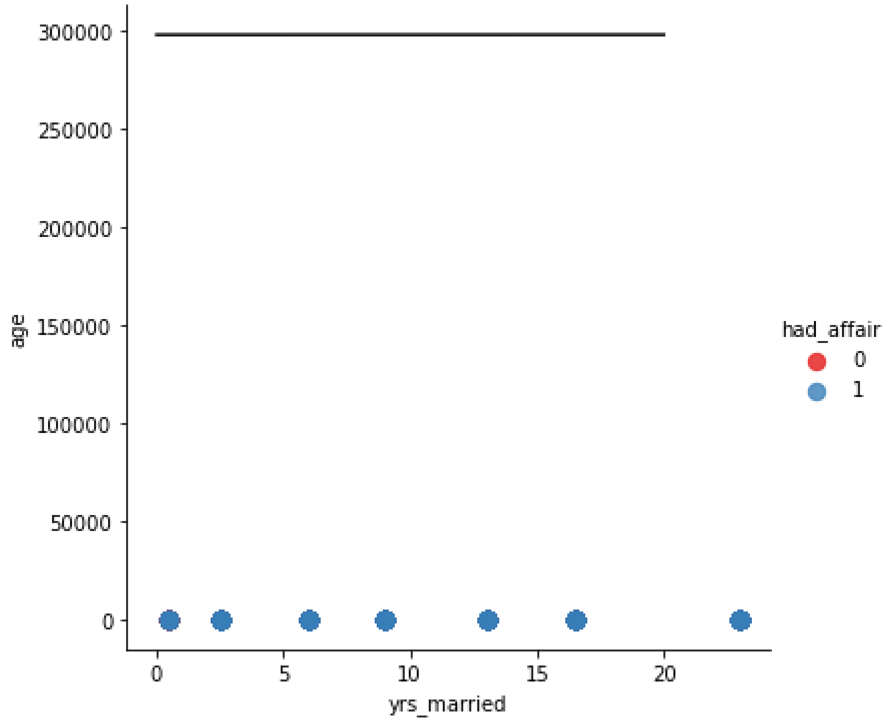
```
SVC(C=8, kernel='linear')
```

In [25]:

```
w = model3.coef_[0]
a = - (w[0] / w[1])
xx = np.linspace(0, 20)
yy = a * xx - ((model2.intercept_[0]) / w[1])
```

In [26]:

```
sns.lmplot('yrs_married', 'age', data=df, hue='had_affair', palette='Set1', fit_reg=False,
plt.plot(xx, yy, color='black');
```



3

In [27]:

```
model4 = svm.SVC(kernel='rbf', gamma=2**-5)
model4.fit(X, Y)
```

Out[27]:

SVC(gamma=0.03125)

In [28]:

```
model5 = svm.SVC(kernel='rbf', gamma=0.1)
model5.fit(X, Y)
```

Out[28]:

SVC(gamma=0.1)

In [29]:

```
model6 = svm.SVC(kernel='rbf', gamma=1)
model6.fit(X, Y)
```

Out[29]:

SVC(gamma=1)

In [30]:

```
model6 = svm.SVC(kernel='rbf', gamma=2)
model6.fit(X, Y)
```

Out[30]:

```
SVC(gamma=2)
```

In [ ]:

```

```

Part B: Now we will try to fit the same dataset with Decision Trees

Follow the standard way of building a model and in particular,

1. Build a Decision Tree Classifier
2. Try using different max_depth = 2, 3, 4 and crierion = 'gini' and 'entropy' to build 6 different models
3. Answer the question of what is your observation from step 2. Does the choice of the criterion important or not. What about max_depth? and What is the best choice of max_depth and criterion
4. Pick 3 models with max_depth = 2, 3, 4 and. You can pick which ever criterions you want and visualize the 3 trees.
5. Build a Random Forest Classifier with, say 100 trees. Comment on its model performance when compared with the individual trees models above

Type your code and answers here

In [36]:

```
from sklearn.model_selection import train_test_split
```

In [37]:

```
df2 = df[["age", "yrs_married", "children", "had_affair"]]
```

In [ ]:

```

```

In [38]:

```
X = df2.drop('had_affair',axis=1)
y = df2['had_affair']
```

In [39]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [40]:

```
from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier(max_depth=3)
model1.fit(X_train,y_train)
```

Out[40]:

```
DecisionTreeClassifier(max_depth=3)
```

In [ ]:

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
predictions = model1.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

2.

In [ ]:

```
model2 = DecisionTreeClassifier(max_depth=2, criterion='gini')
model2.fit(X_train,y_train)
predictions = model2.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

In [ ]:

```
model3 = DecisionTreeClassifier(max_depth=2, criterion='entropy')
model3.fit(X_train,y_train)
predictions = model3.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

In [ ]:

```
model4 = DecisionTreeClassifier(max_depth=3, criterion='gini')
model4.fit(X_train,y_train)
predictions = model4.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

In [69]:

```
model5 = DecisionTreeClassifier(max_depth=3, criterion='entropy')
model5.fit(X_train,y_train)
predictions = model5.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[1299    1]
 [ 609    1]]
              precision    recall  f1-score   support

           0       0.68      1.00      0.81      1300
           1       0.50      0.00      0.00       610

    accuracy                           0.68      1910
   macro avg       0.59      0.50      0.41      1910
weighted avg       0.62      0.68      0.55      1910

0.680628272251309
```

In [70]:

```
model6 = DecisionTreeClassifier(max_depth=4, criterion='entropy')
model6.fit(X_train,y_train)
predictions = model6.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[1278   22]
 [ 586   24]]
              precision    recall  f1-score   support

           0       0.69      0.98      0.81      1300
           1       0.52      0.04      0.07       610

    accuracy                           0.68      1910
   macro avg       0.60      0.51      0.44      1910
weighted avg       0.63      0.68      0.57      1910

0.6816753926701571
```

In [71]:

```
model7 = DecisionTreeClassifier(max_depth=4, criterion='gini')
model7.fit(X_train,y_train)
predictions = model7.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[1298    2]
 [ 609    1]]
              precision    recall  f1-score   support

           0       0.68      1.00      0.81      1300
           1       0.33      0.00      0.00       610

    accuracy                           0.68      1910
   macro avg       0.51      0.50      0.41      1910
weighted avg       0.57      0.68      0.55      1910

0.6801047120418848
```

3

Answer the question of what is your observation from step 2. Does the choice of the criterion important or not. What about max_depth? and What is the best choice of max_depth and criterion

choice is not important and max_depth not influence result. The best choice of max_depth is 2 and criterion are ''entropy' and'gini 'when max_depth is 2 because after running ten times, the first group has the highest probability of having the highest value

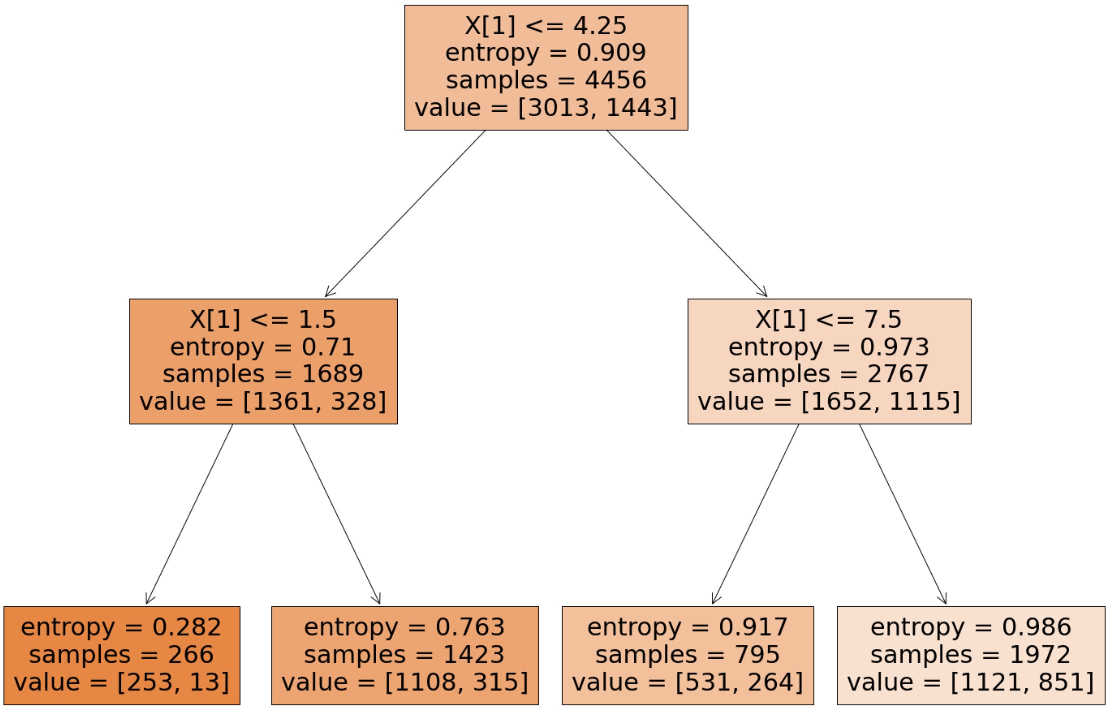
4

In [ ]:

```

```

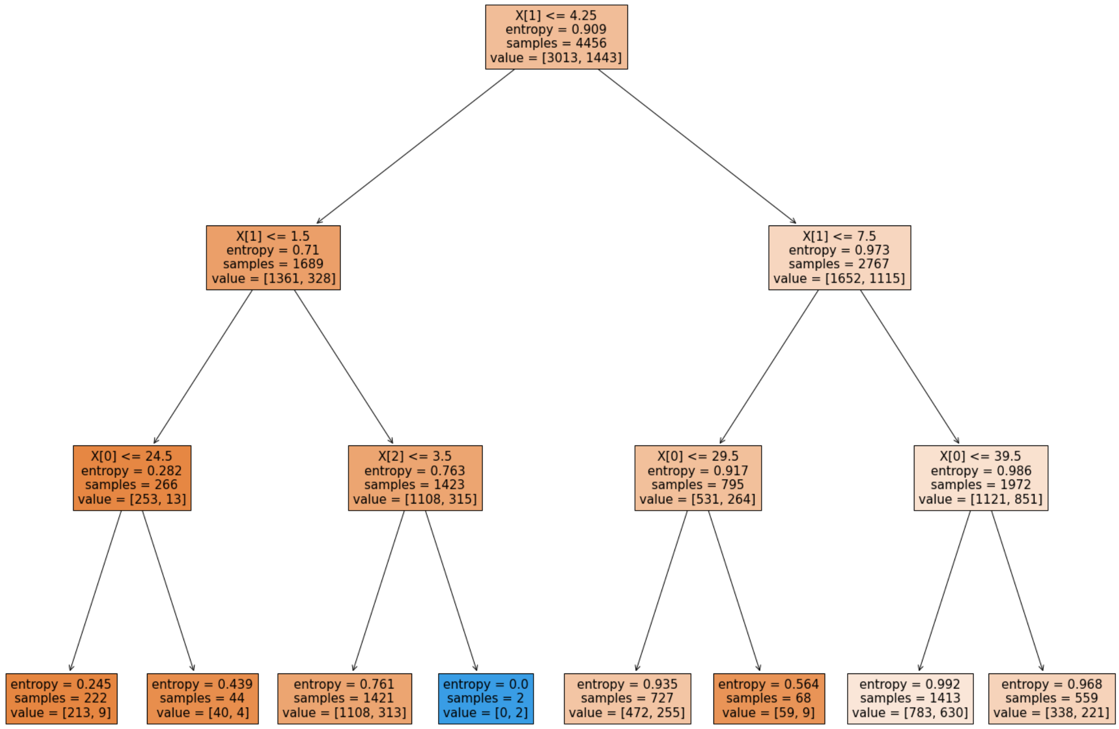
In [72]:

```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model3, filled=True)
```



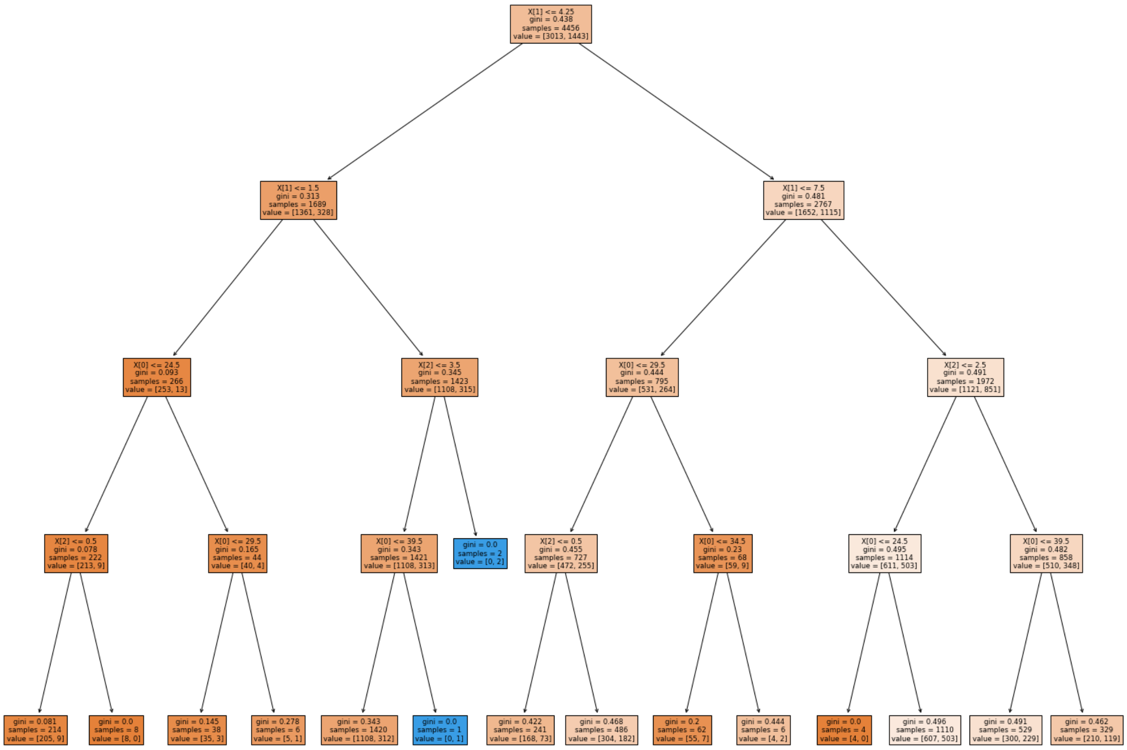
In [73]:

```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model5, filled=True)
```



In [74]:

```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model7, filled=True)
```



5.Build a Random Forest Classifier with, say 100 trees. Comment on its model performance when compared with the individual trees models above

Random is not good for individual trees models because random model values is always lower than individual trees models

In [75]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

Out[75]:

```
RandomForestClassifier()
```

In [76]:

```
rfc_pred = rfc.predict(X_test)
print(classification_report(y_test, rfc_pred))
print(accuracy_score(y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.93      0.80      1300
           1       0.47      0.13      0.21       610

    accuracy                           0.68      1910
   macro avg       0.58      0.53      0.50      1910
weighted avg       0.62      0.68      0.61      1910

0.675392670157068
```

In [ ]:

```

```

Part C: Now finally create a dataframe including all other categorical variable and build decision tree model

In [77]:

```

df4 = pd.concat([df2, rate_marriage, religious, edu, husb_occ, occ], axis=1)
df4.columns
```

Out[77]:

```
Index([        'age', 'yrs_married',    'children',  'had_affair',
             'rate1',       'rate2',       'rate3',       'rate4',
              'rel1',        'rel2',        'rel3',          12.0,
                14.0,          16.0,          17.0,          20.0,
                 2.0,           3.0,           4.0,           5.0,
                 6.0,        'occ2',        'occ3',        'occ4',
              'occ5',        'occ6'],
      dtype='object')
```

Use the same model as in Part B step 1 with this new dataframe. Comment on whether the additional variables help the model performance or not

Yes, a little help.

In [78]:

```
X = df4.drop('had_affair',axis=1)
y = df4['had_affair']
```

In [79]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

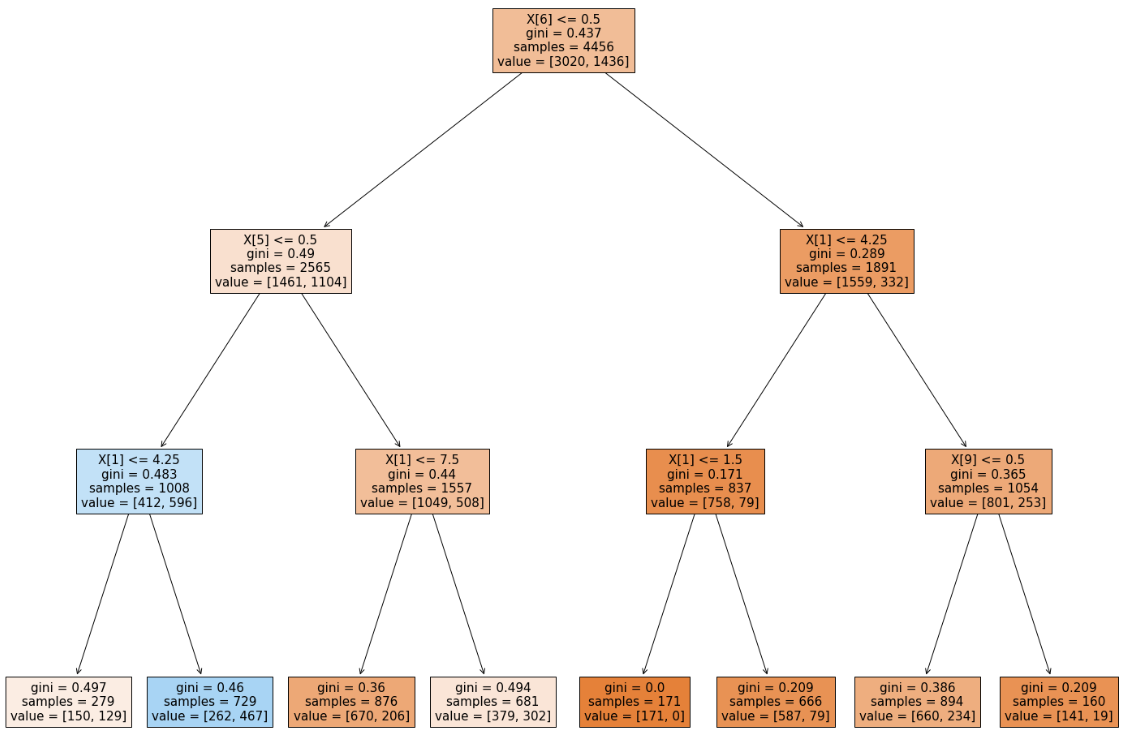
In [80]:

```
model10 = DecisionTreeClassifier(max_depth=3)
model10.fit(X_train,y_train)
predictions = model10.predict(X_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
print(accuracy_score(y_test, predictions))
```

```
[[1168  125]
 [ 418  199]]
              precision    recall  f1-score   support

           0       0.74      0.90      0.81      1293
           1       0.61      0.32      0.42       617

    accuracy                           0.72      1910
   macro avg       0.68      0.61      0.62      1910
weighted avg       0.70      0.72      0.69      1910

0.7157068062827225
```

In [81]:

```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(model10, filled=True)
```



In [82]:

```
# Type your code here, fill in the missing code here

edu.columns = ['edu1','edu2','edu3','edu4','edu5' ]
# ...

# df2 = pd.concat[df, ....]

# df2.columns
```

Use the same model as in Part B step 1 with this new dataframe. Comment on whether the additional variables help the model performance or not

Type your code and answers here

In [ ]:

```

```

In [ ]:

```

```