In [1]:
```
#Importing necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, confusior
```

In [2]:
```
#Loading data
bank = pd.read_csv("bank.csv",sep=';')
bank.head(2)
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_we
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	m
1	57	services	married	high.school	unknown	no	no	telephone	may	m

2 rows × 21 columns

In [3]:
```
bank.describe()
```

Out[3]:

	age	duration	campaign	pdays	previous	emp.var.rate	cc
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	4
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	

In [4]:
```
#Checking Missing values
bank.isnull().sum()
```

Out[4]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

In [5]:
```
#Checking for duplicates
bank.duplicated().sum()
```

Out[5]:
```
12
```

In [6]:
```
#Investigating these 12 duplicates
bank[bank.duplicated()]
```

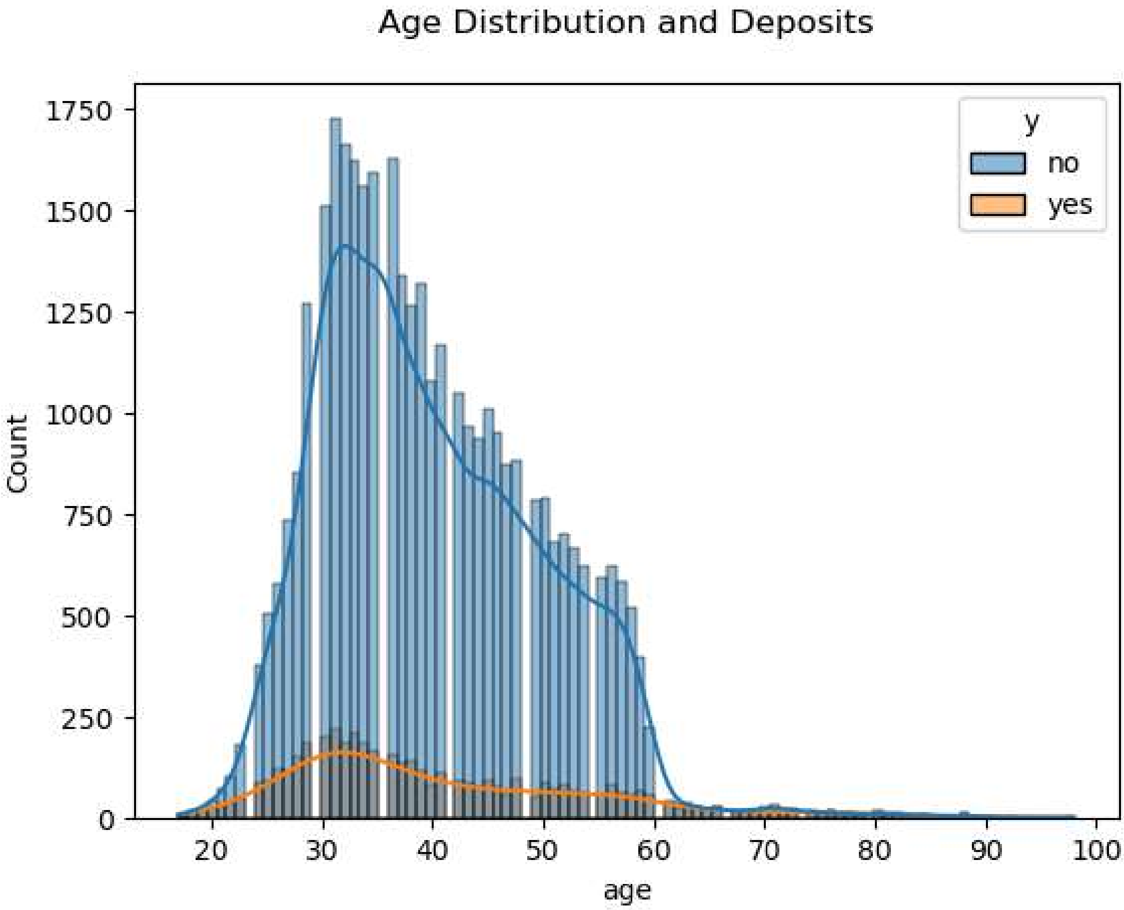
Out[6]:

	age	job	marital	education	default	housing	loan	contact	month
1266	39	blue-collar	married	basic.6y	no	no	no	telephone	may
12261	36	retired	married	unknown	no	no	no	telephone	jul
14234	27	technician	single	professional.course	no	no	no	cellular	jul
16956	47	technician	divorced	high.school	no	yes	no	cellular	jul
18465	32	technician	single	professional.course	no	yes	no	cellular	jul
20216	55	services	married	high.school	unknown	no	no	cellular	aug
20534	41	technician	married	professional.course	no	yes	no	cellular	aug
25217	39	admin.	married	university.degree	no	no	no	cellular	nov
28477	24	services	single	high.school	no	yes	no	cellular	apr
32516	35	admin.	married	university.degree	no	yes	no	cellular	may
36951	45	admin.	married	university.degree	no	no	no	cellular	jul
38281	71	retired	single	university.degree	no	no	no	telephone	oct

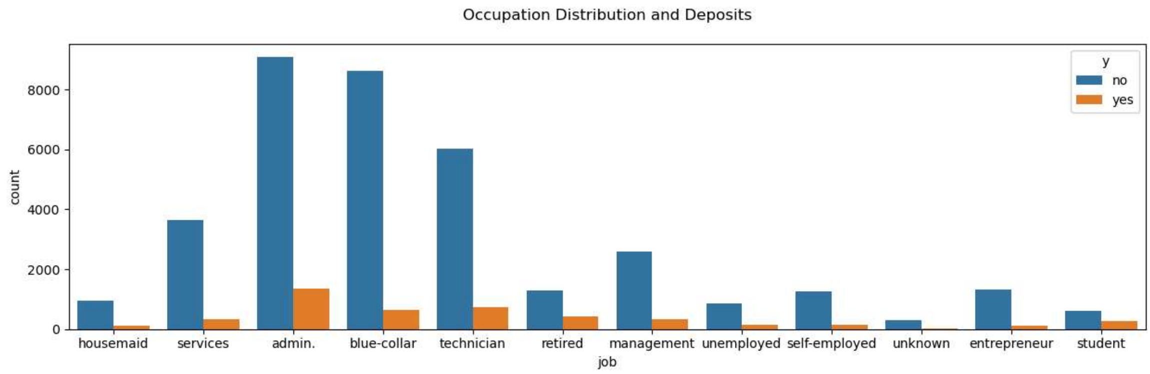
12 rows × 21 columns

Age Distribution

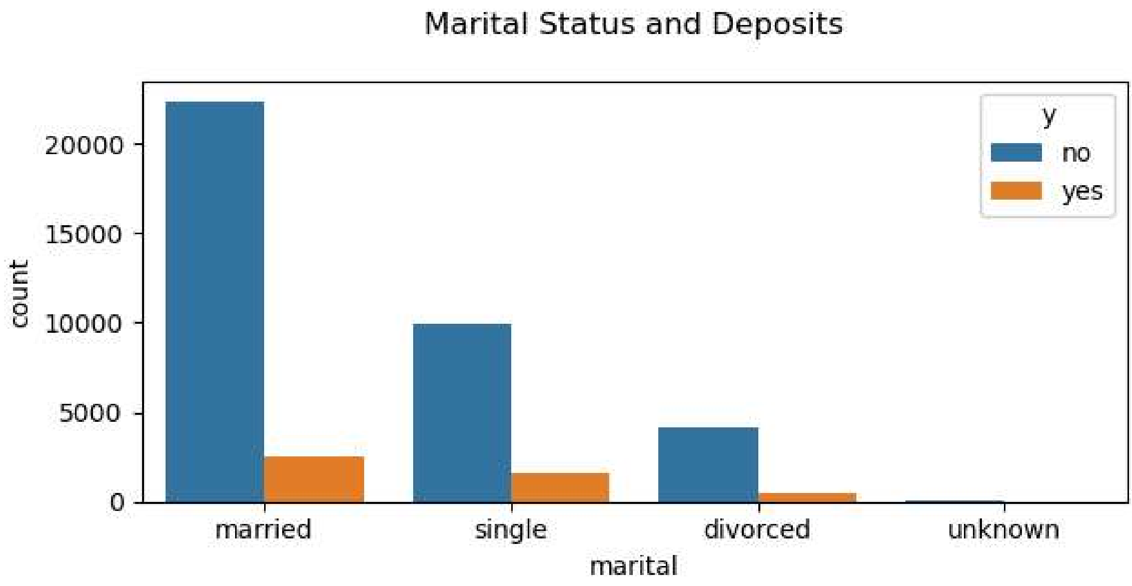
In [7]:
```
sns.histplot(x="age", data=bank, kde=True, hue= "y")
plt.title("Age Distribution and Deposits\n")
plt.show()
```



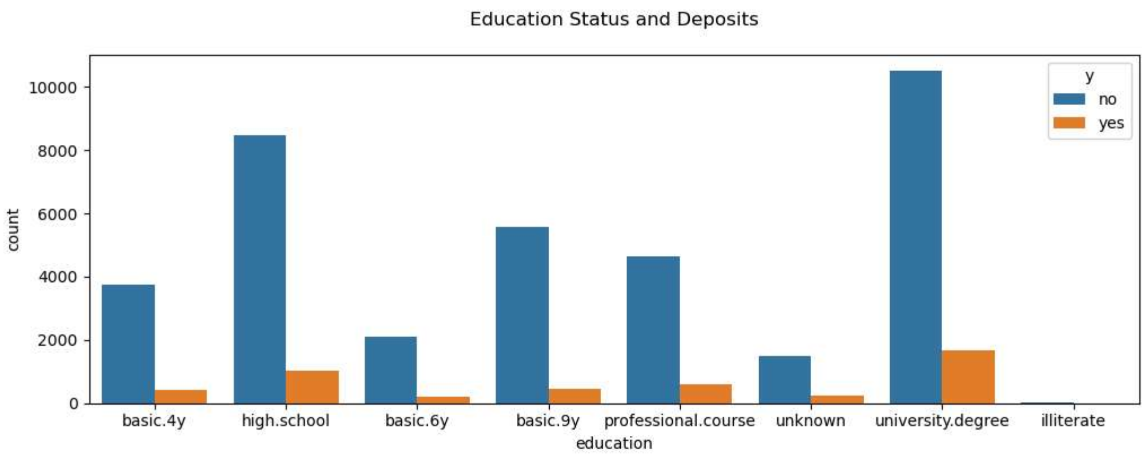
In [8]:
```
#Distribution of Occupation
plt.figure(figsize=(15,4))
sns.countplot(x="job", data= bank, hue ="y")
plt.title("Occupation Distribution and Deposits\n")
plt.show()
```



In [9]:
```
# Distribution of Marital Status
plt.figure(figsize=(7,3))
sns.countplot(x="marital", data= bank, hue ="y")
plt.title("Marital Status and Deposits\n")
plt.show()
```



In [10]:
```
# Distribution of Education Status
plt.figure(figsize=(12,4))
sns.countplot(x="education", data= bank, hue ="y")
plt.title("Education Status and Deposits\n")
plt.show()
```



In [11]:
```
# Label Encoding Categorical Features
cols = bank.select_dtypes("object").columns
cols
```

Out[11]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'conta
ct',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

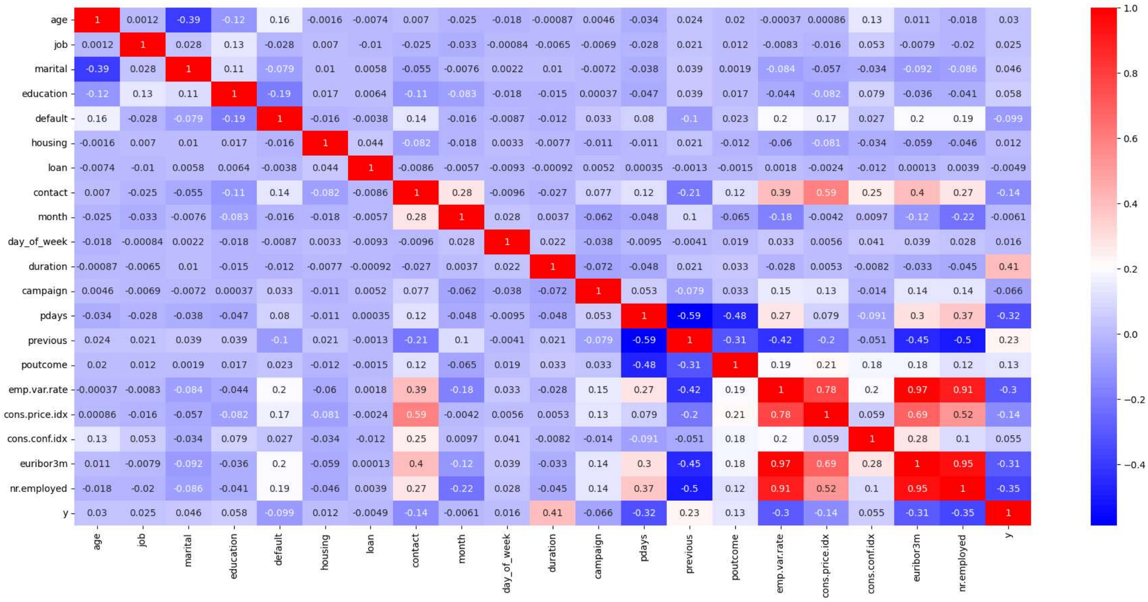
In [12]:
```
le = LabelEncoder()

bank[cols] = bank[cols].apply(le.fit_transform)
bank.head(3)
```

Out[12]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	car
0	56	3	1	0	0	0	0	1	6	1	...	
1	57	7	1	3	1	0	0	1	6	1	...	
2	37	7	1	3	0	2	0	1	6	1	...	

3 rows × 21 columns

In [13]:
```
# Correlation Analysis using Heatmap
plt.figure(figsize=(23,10))
sns.heatmap(bank.corr(), cmap='bwr', annot=True)
plt.show()
```



In [ ]:
```
# Outcome "y" is positively correlated with duration of call and also shows
# multicolinearty can be seen among some
# input features. This can be handled by dropping those variables or by perf
```

In [14]:
```
#Standardisation
#Splitting input and output
X = bank.drop("y", axis=1)
y = bank.y
scaler = StandardScaler()

X_scaled = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
```

In [16]:
```
# Model building - Decision Tree Classifier
#Train-test split
train_X, test_X, train_y, test_y = train_test_split(X_scaled, y, test_size=0
decision_tree = DecisionTreeClassifier()
decision_tree.fit(train_X, train_y)
```

Out[16]:
```
DecisionTreeClassifier()
```

In [18]:
```
print('Train Score: {}'.format(decision_tree.score(train_X, train_y)))
print('Test Score: {}'.format(decision_tree.score(test_X, test_y)))
```

```
Train Score: 1.0
Test Score: 0.8883224083515416
```

In [19]:
```
cross_val_score(decision_tree, train_X, train_y, cv=5).mean()
```

Out[19]:
```
0.8880371720376579
```

In [20]:
```
ypred = decision_tree.predict(test_X)
print(classification_report(test_y,ypred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     10968
           1       0.50      0.51      0.51      1389

    accuracy                           0.89     12357
   macro avg       0.72      0.72      0.72     12357
weighted avg       0.89      0.89      0.89     12357
```

In [21]:
```
# Hyperparameter tunning
#Applying Grid search cv to find best estimaters to improve model performanc

param_grid = {
    'max_depth': [3, 5, 7,10, None],
    'criterion' : ['gini', 'entropy'],
    'min_samples_leaf': [3, 5, 7, 9,10,20]
    }
```

In [22]:
```
gscv = GridSearchCV(decision_tree, param_grid, cv=5, verbose=1)
gscv.fit(train_X, train_y)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
```

Out[22]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, 10, None],
                         'min_samples_leaf': [3, 5, 7, 9, 10, 20]},
             verbose=1)
```

In [23]:
```
gscv.best_params_
```

Out[23]:
```
{'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 10}
```

In [24]:
```
gscv.best_estimator_
```
Out[24]:
```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=10)
```

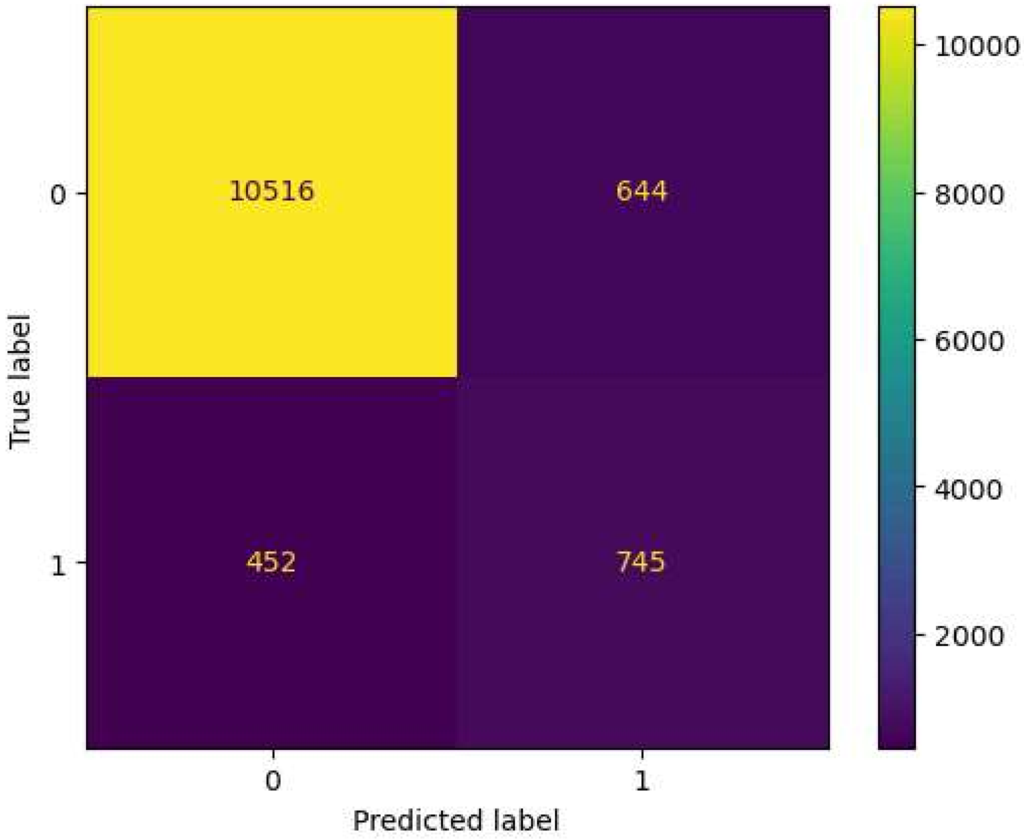
In [25]:
```
cross_val_score(gscv.best_estimator_, train_X, train_y, cv=5).mean()
```
Out[25]:
```
0.914258828247674
```

In [26]:
```
clf = DecisionTreeClassifier(criterion= 'gini', max_depth= 5, min_samples_le
clf.fit(train_X, train_y)
```
Out[26]:
```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=3)
```

In [27]:
```
print('Train Score: {}'.format(clf.score(train_X, train_y)))
print('Test Score: {}'.format(clf.score(test_X, test_y)))
```
```
Train Score: 0.9177274461517116
Test Score: 0.9113053330096301
```

In [28]:
```
pred_y = clf.predict(test_X)
```

In [29]:
```
#Confusion Matrix
cm = confusion_matrix(pred_y, test_y)
ConfusionMatrixDisplay(cm, display_labels=clf.classes_).plot()
plt.show()
```



In [30]:
```
#Classification Report
print(classification_report(pred_y, test_y))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     11160
           1       0.54      0.62      0.58      1197

    accuracy                           0.91     12357
   macro avg       0.75      0.78      0.76     12357
weighted avg       0.92      0.91      0.91     12357
```

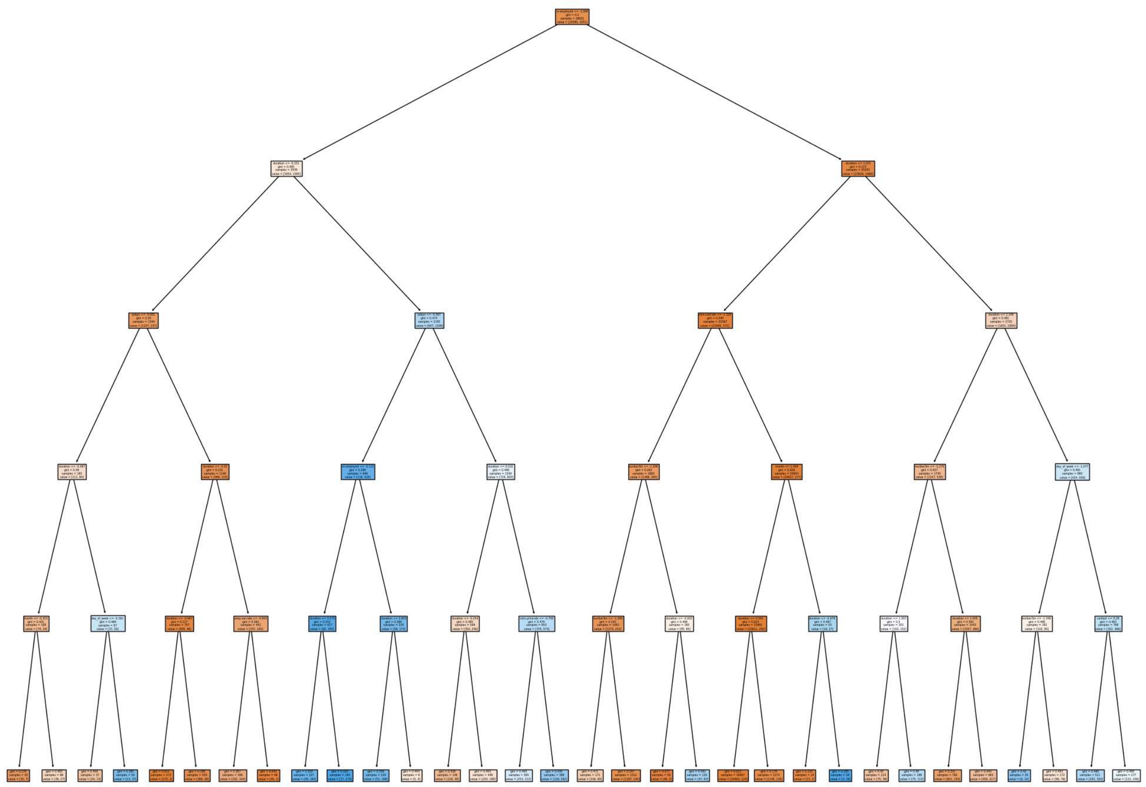
In [31]:
```
#Accuracy Score
accuracy = accuracy_score(test_y,pred_y)
print("Test Accuracy of Decision Tree Classifier : {}".format(accuracy*100))
```

```
Test Accuracy of Decision Tree Classifier : 91.13053330096301
```

In [32]:
```
#Cross Validation Score
Cross_val = cross_val_score(clf, test_X,test_y, cv=5).mean()
print("Cross-Validation Accuracy Scores Decision Tree : ",Cross_val*100)
```

```
Cross-Validation Accuracy Scores Decision Tree :  91.082010873053
```

In [33]:
```
from sklearn import tree
fig = plt.figure(figsize=(25,20))
t= tree.plot_tree(clf,filled=True,feature_names=X.columns)
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: