Results

We decided to use a decision tree model to classify whether or not a person had a stroke based on select key variables. Our first objective was to create a cleaned training dataset that contained all numeric and dummy variables.

In [6]:
```
df_train.head()
```

Out[6]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_statu
0	1	36.0	0	0	1	0	0	65.87	32.2	
1	0	45.0	0	0	1	0	1	68.66	25.3	
2	1	58.0	0	0	0	1	1	170.93	30.7	
3	0	61.0	0	0	1	1	0	69.88	27.1	
4	0	78.0	0	0	0	2	0	103.86	30.6	

From here we proceeded to clean the testing dataset, however, we noticed that the BMI variable still had 42 missing values.

In [8]:
```
#Clean the test set
df_test['ever_married'] = df_test['ever_married'].replace(['Yes','No'],[1,0])
df_test['work_type'] = df_test['work_type'].replace(['Govt_job','Self-employed','Private','children','Never_worke
df_test['Residence_type'] = df_test['Residence_type'].replace(['Urban','Rural'],[0,1])
df_test['smoking_status'] = df_test['smoking_status'].replace(['formerly smoked','never smoked','Unknown','smokes
df_test['gender'] = df_test['gender'].replace(['Male','Female','Other'],[1,0,2])
df_test = df_test.drop(['Unnamed: 0','id'],axis=1)
df_test['bmi'].isna().sum()
```

Out[8]: 42

We went ahead and dropped these values since 42 is a small portion of the approximate 1000 values obtained from the testing set.

In [9]:
```
df_test = df_test.dropna(subset=['bmi'])
```

Next, we isolated our target variable– stroke–and got rid of variables that we deemed unnecessary to predict if someone had a stroke or not. The specific variables we dropped from the train and test sets were “stroke” (our target variable), “ever_married”, and “residence_type”. We then normalized the “age”, “avg_glucose_level”, and “bmi” variables to have a better-matched correlation. **NOTE: The rest of the variables are dummy variables consisting of 0, 1, 2, and 3; thus we didn’t run the normalization function on those **

In [10]:
```
y_train = df_train['stroke']
X_train = df_train.drop('stroke',axis=1)
y_test = df_test['stroke']
X_test = df_test.drop('stroke',axis=1)

X_train['bmi'] = X_train['bmi'].fillna(X_train['bmi'].mean())
X_test['bmi'] = X_test['bmi'].fillna(X_test['bmi'].mean())
```

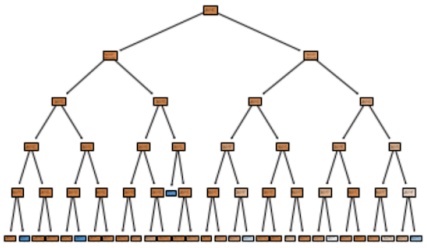
In [96]:
```
def maxmin(x):
    x = (x-min(x))/(max(x)-min(x))
    return x
#First run we are using gender, age, hypertension, heart disease, glucose level, bmi, smoking status
X_train = df_train.drop(['stroke','ever_married','Residence_type'],axis=1)
X_test = df_test.drop(['stroke','ever_married','Residence_type'],axis=1)
#Normalize numeric variables
X_train[['age','avg_glucose_level','bmi']] = X_train[['age','avg_glucose_level','bmi']].apply(maxmin)
X_test[['age','avg_glucose_level','bmi']] = X_test[['age','avg_glucose_level','bmi']].apply(maxmin)
X_train.head()
```

Out[96]:

	gender	age	hypertension	heart_disease	work_type	avg_glucose_level	bmi	smoking_status
0	1	0.438477	0	0	0	0.049626	0.250859	0
1	0	0.548340	0	0	0	0.062506	0.171821	1
2	1	0.707031	0	0	1	0.534623	0.233677	2
3	0	0.743652	0	0	1	0.068138	0.192440	1
4	0	0.951172	0	0	2	0.225002	0.232532	2

At this point we were ready to fit our decision tree model. To make the data as consistent as possible, we varied the max_depth and random_state variables to see which provided the best results. We ended up going with max_depth=5 and random_state=10.

In [97]:
```
#Run decision tree model
from sklearn import tree
import matplotlib.pyplot as plt
model = tree.DecisionTreeClassifier(max_depth=5,random_state=10)
cart = model.fit(X_train,y_train)
tree.plot_tree(cart,filled=True)
plt.show()
```



Afterward, we created a confusion matrix to see how accurate our model was at predicting who would have a stroke and the results were very surprising.

In [98]:
```
#Make predictions and crosstab
y_hat = cart.predict(X_test)
pd.crosstab(y_test,y_hat)
```

Out[98]:

col_0	0	1
stroke		
0	942	1
1	38	0

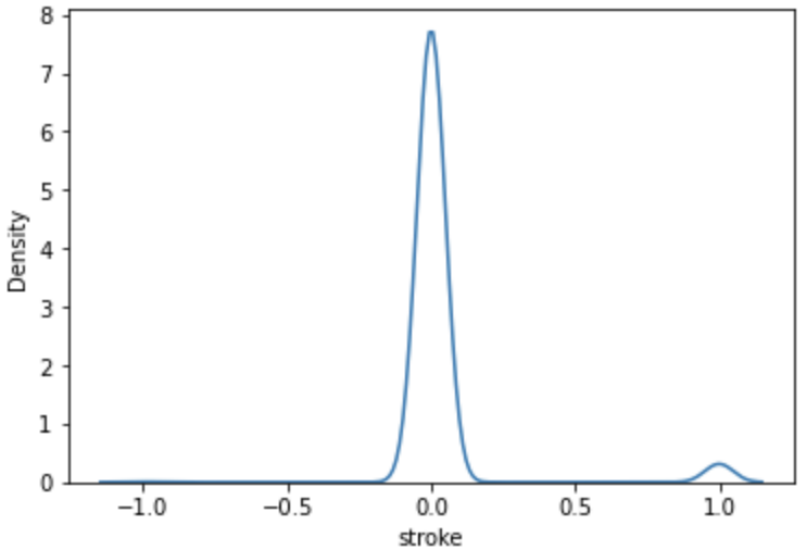
Somehow, our model was excellent at predicting people who have never had a stroke, and terrible at predicting people who did have a stroke. For people who never had a stroke, our model correctly predicted 942/943 which is 99.9% correct. However, for people who had a stroke, our model predicted 0/38 which is 0% correct. With this in mind, we wanted to see the overall accuracy score of our model and it turned out to be 0.9602, which was pretty solid. Our RMSE value for the model was 0.199.

In [99]:
```
#Calculate SSE
SSE = np.sum((y_test-y_hat)**2)
#accuracy score
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_hat))
#RMSE
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test,y_hat,squared=False))
```

```
0.9602446483180428
0.19938744113398213
```

We ran a residuals plot as well to visualize our prediction and we can see that the model was far more accurate at predicting 0 (no stroke) than 1 (stroke)

In [100…
```
residuals = y_test - y_hat
import seaborn as sns
sns.kdeplot(residuals)
plt.show()
```



We then repeated the decision tree model on a new dataset that excluded “age”, “gender”, and “work_type” to see if these variables were necessary for creating an accurate model that predicts whether or not someone had a stroke before.

```
X_test_2.head()
```

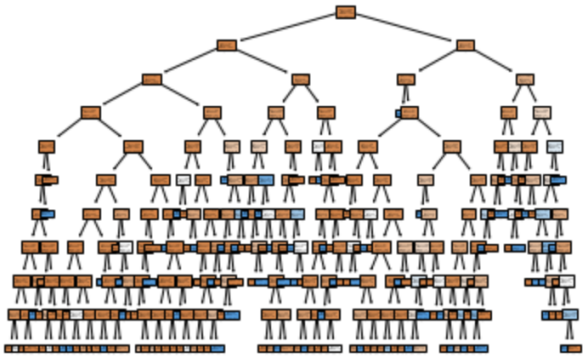
Out[52]:

	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status
0	1	0	0.157026	0.519700	1
1	0	0	0.267102	0.114447	2
2	0	0	0.066199	0.457786	0
3	0	0	0.747444	0.724203	2
4	1	0	0.722913	0.808630	1

We once again ran the decision tree model on this dataset.

In [33]:

```
#Run decision tree model
from sklearn import tree
import matplotlib.pyplot as plt
model = tree.DecisionTreeClassifier(max_depth=10,random_state=10)
cart = model.fit(X_train_2,y_train)
tree.plot_tree(cart,filled=True)
plt.show()
```



Like before, we ran a confusion matrix on this dataset, and interestingly, the accuracy for predicting no stroke went down slightly, but the accuracy for predicting stroke correctly went up. For people who never had a stroke, our model correctly predicted 926/943 which is 98.2% correct. However, for people who had a stroke, our model predicted 4/38 which is 10.5% correct

Out[34]:

col_0	0	1
stroke		
0	926	17
1	34	4

To test overall accuracy, we calculated the accuracy score and RMSE and got lower values than the original model.

In [35]:

```
#Calculate SSE
SSE = np.sum((y_test-y_hat)**2)
#accuracy score
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_hat))
#RMSE
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test,y_hat,squared=False))
```

```
0.9480122324159022
0.22800826209613076
```

Thus, we came to the conclusion that our original model was the best model to use since it produced the lowest RMSE score at 0.199.