Summary

The question we are trying to answer through data analysis is predicting the likelihood that a person will have a stroke. We were provided a wide variety of data but decided to focus on the key influential variables in our opinion: smoking status, average glucose level, age, hypertension, heart disease, BMI, gender, and type of work. We omitted variables such as marital status and residence type from our analysis because we believed that these variables do not have as significant of an impact on whether a person has had a stroke. Although k nearest neighbor was an intriguing means of data analysis to pursue, we have decided to use decision trees to examine the data instead. We made this decision because of the straightforwardness of decision trees, and our group members are more comfortable with them. After conducting our analysis, we found that our model was very effective at predicting whether people will not have a stroke, but awful at predicting whether people did have a stroke. Our model predicted with approximately 99.90% accuracy if people did not have a stroke, correctly predicting 942 out of 943. On the other hand, for people who had a stroke, our model predicted with 0% accuracy, getting 0 out of 38 cases correct. Knowing that the success of the predictions of our model varies drastically depending on whether it is handling someone who had a stroke or did not have a stroke, we decided to calculate the overall accuracy score, yielding a value of 0.9602, which is fairly impressive. Our RMSE value for the model was 0.199. Due to the peculiarity of our model's accuracy, we decided to try a more narrow set of predictor variables, excluding the type of work, age, and gender of the patient. Although this model slightly reduced the accuracy of predicting for people who have never had a stroke, dropping to 98.2% (926 out of 943 cases) accuracy, the accuracy of predicting for people who have had a stroke increased from 0% to 10.5% (4 out of 38 cases). However, after computing the overall accuracy score and RMSE of our new model, 0.9480 and 0.2280, respectively, we decided to stick with our original model as it is our most accurate model.

Data

The dataset used for this project consists of relevant information for determining a patient's likelihood to experience a stroke (and if a patient experiences a stroke). Upon importing the dataset, we cleaned it by converting all non-binary and categorical variables into binary values. We used smoking_status, avg_glucose_level, age, bmi, hypertension, heart_disease work_type, and gender as the key predictors in our model to determine whether or not a patient experiences a stroke. We chose most of these variables based on the obvious health relations with stroke likelihood. After adjusting our model and running it with several other combinations for predictors, we found that work_type and gender surprisingly made our model more accurate. The table below lists and summarizes all of the relevant columns in our analysis, as well as our transformations to each variable.

Variable name	**Description**	**Variable values (original == transformed)**

smoking_status	Categorizes patients based on whether or not the respondent smokes, considers both historically and currently	formerly smoked == 0 never smoked == 1 Unknown == 2 smokes == 3
avg_glucose_level	Lists the blood sugar level of each patient in numeric form	Unchanged
age	Patient age in numeric form	Unchanged
bmi	Body mass index of each respondent in numeric form	Unchanged
hypertension	Categorizes patients on whether or not they have hypertension, binary	Unchanged (0 is does not have hypertension and 1 is has hypertension)
heart_disease	Categorizes patients on whether or not they have heart disease, binary	Unchanged (0 is does not have heart disease and 1 is has heart disease)
gender	Categorizes patients based on gender	Female == 0 Male == 1 Other == 2
work_type	Patient's employment type	0 == govt_job 1 == self_employed 2 == private 3 == children 4 == never_worked
stroke	Whether or not the patient suffered a stroke during the same period	Unchanged (0 is did not experience a stroke and 1 is did experience a stroke)

When looking at the BMI variable, we found 159 NaNs when computing value counts. We decided to drop the NaNs because we considered 159 out of 4087 rows a non-significant amount of data to drop since we ended up with 3928 rows, which was still sufficient for our modeling purposes.

We had some challenges with understanding the work_type data. Since there was no proper codebook provided for this dataset, we were unclear what "children" meant. Based on context and our understanding, we figured that this was meant for stay-at-home parents. We had to clean the data by creating binaries/numerical representations of the column values. For example, continuing with work_type, the responses were “govt_job,” “self_employed,” “private,” “children,” and “never_worked.” To prepare for our analysis, we converted those values to 0, 1, 2, 3, and 4, respectively. We applied this process to other categorical variables used in our model.

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]:
```python
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())
```

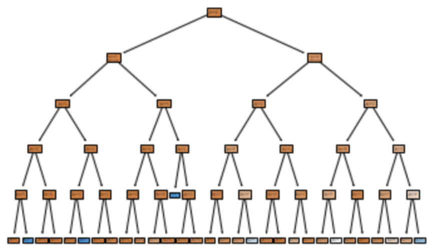
In [96]:
```python
def maxmin(x):
    x = (x-min(x))/(max(x)-min(x))
    return x
#First run we are using gender, age, hypertension, heart disease, glucose level, bmi, smoking status
X_train = df_train.drop(['stroke','ever_married','Residence_type'],axis=1)
X_test = df_test.drop(['stroke','ever_married','Residence_type'],axis=1)
#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 set a random state. We varied the max_depth parameter to see which provided the best results. A max_depth of 5 resulted in the most accurate model.

In [97]:
```python
#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. The results were very surprising.

In [98]:
```python
#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

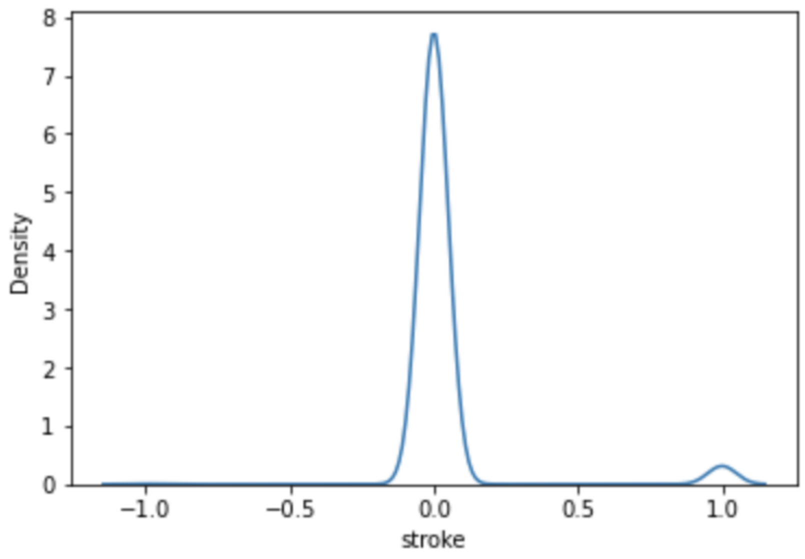
Our model was excellent at predicting people who have never had a stroke, and terrible at predicting people who did have a stroke. For people who never had a stroke, our model correctly predicted 942/943 which is 99.9% correct. However, for people who had a stroke, our model predicted 0/38 which is 0% 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.

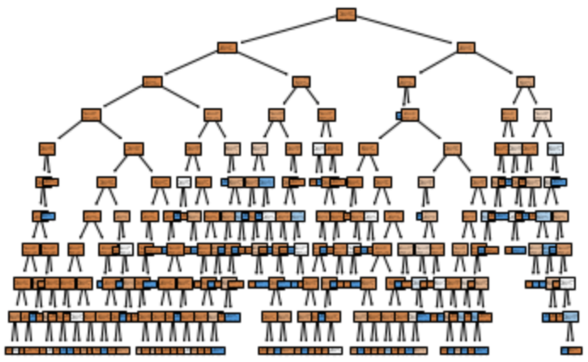
```
X_test_2.head()
```

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.

Conclusion

In this project, we used the decision tree model to predict the likelihood of a person having a stroke. We used variables like smoking status, average glucose level, age, hypertension, heart disease, BMI, gender, and type of work, while dropping variables like marital status and residence type. We transformed some variables into dummy variables, with their categories represented numerically. To clean further, we also dropped any observations with NAs in the variables we were testing, resulting in about 1000 observations. When testing our model, we came to the conclusion that it had 99.9% accuracy at predicting if people did not have a stroke, but 0% accuracy at predicting those who did have a stroke. Our overall accuracy score of the model was 0.9602, and the RMSE value was 0.199. A residual plot to visualize our predictions also showed that predicting a value of 0 for stroke was much higher than predicting a value of 1. We decided to repeat testing, but excluding the variables representing age, gender, and work type. On this second model, it had an accuracy of 98.2 for those who had never had a stroke, and 10.5% for those who did. However, this model had a lower accuracy score of 0.9480 and a higher RMSE value of 0.228. With this information, we concluded that our first model was the better of the two.

Considering our choices of dropping variables, in order to make this project more comprehensive, we could have run models where we had kept marital status and residence type. Although we dropped these initially because we thought they had been more irrelevant than others, it would not hurt to test them as well, as it could yield surprising results.
Although our choice to use a decision tree model was mostly because of ease of interpretability, they might not capture complex relationships as effectively as other models. There are a lot of possible variables that go into the likelihood of a stroke, and our model may not have been able to convey that as well, which is why it struggled with positive stroke cases.

For future reference, it could benefit us to do more exhaustive testing among the variables and compare the results of different combinations, in order to get more accurate results, especially in the instance of positive stroke cases. Additionally, although decision trees allow for understandability, it is important to balance with predictive power, potentially trying out different models, like k nearest neighbor or linear models.