Decision Tree Regressor

Decision tree regressors use decision trees to develop a regression model capable of predicting continous a variable using other variables. My goal is to predict the 'fastest lap speed' of Formular 1 data using the following variables: points, laps, altitude, fastestLapDuration, averagePitstopDuration, and averageLapDuration. I will compare the results from this model with the results from the best multiple regression model model obtained to see which model suits this analysis better.

In [9]:
```python
# Setup

import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
```

In [4]:
```python
# Read the data
f1_full_df = pd.read_csv("Formula_One.csv")

# View first several rows
f1_full_df.head()
```

Out[4]:

	raceId	driverId	circuitId	grid	finishingPosition	points	laps	raceDuration	rank	fastestLapSpeed	altitude	averagePitstopDuration	avera
0	841	20	1	1	1	25	58	5370259	4	212.488	10	23319.500000	
1	841	1	1	2	2	18	58	5392556	8	211.382	10	23213.000000	
2	841	808	1	6	3	15	58	5400819	7	211.969	10	25109.000000	
3	841	4	1	5	4	12	58	5402031	2	213.336	10	24055.000000	
4	841	17	1	3	5	10	58	5408430	3	213.066	10	24058.666667	

In [5]:
```python
# Subset data with the variables from the best model using Linear Regression

f1_df = f1_full_df[['points', 'laps', 'altitude', 'fastestLapDuration', 'averagePitstopDuration', 'averageLapDu
                    'fastestLapSpeed']]

f1_df.head()
```

Out[5]:

	points	laps	altitude	fastestLapDuration	averagePitstopDuration	averageLapDuration	fastestLapSpeed
0	25	58	10	89844	23319.500000	92590.672414	212.488
1	18	58	10	90314	23213.000000	92975.103448	211.382
2	15	58	10	90064	25109.000000	93117.568966	211.969
3	12	58	10	89487	24055.000000	93138.465517	213.336
4	10	58	10	89600	24058.666667	93248.793103	213.066

Decision Tree Regressor

Using DecisionTreeRegressor from sklearn.tree to predict the fastestLapSpeed

I use variables from the best model selected from linear regression for this procedure.

a) Without specifying the maxdepth

In [31]:
```python
# Split the data into predictor variables and the target variable
X = f1_df[['points', 'laps', 'altitude', 'fastestLapDuration', 'averagePitstopDuration', 'averageLapDuration']]
y = f1_df['fastestLapSpeed']

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Fit the model to the training data
model1 = DecisionTreeRegressor(random_state=0)
model1.fit(X_train, y_train)

# Predict the target variable using the test data
y_pred1 = model1.predict(X_test)

# Evaluate the model using metrics such as R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE)
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print('R-squared:', r2_score(y_test, y_pred1))
print('MSE:', mean_squared_error(y_test, y_pred1))
print('MAE:', mean_absolute_error(y_test, y_pred1))
```

```
R-squared: 0.9973022539746974
MSE: 1.2356288946540885
MAE: 0.4850330188679239
```

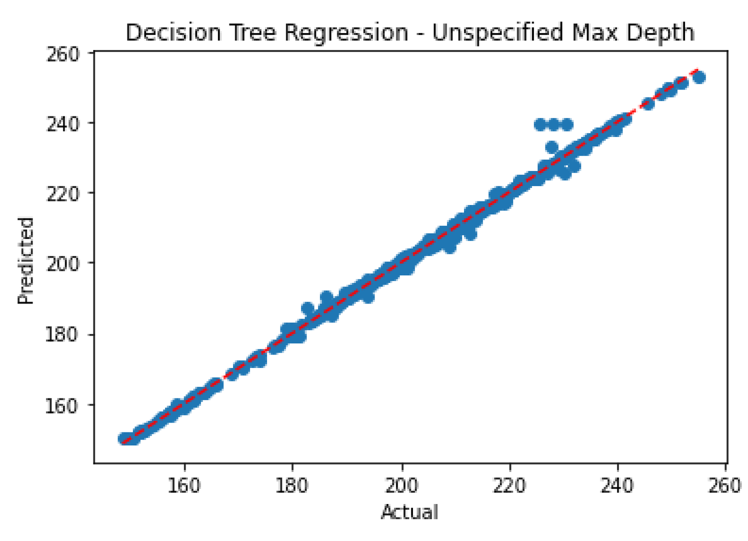
The R squared is so high at 0.9973. The model could be overfitting the data.

In [27]:
```python
# Checking model1 depth
model1.get_depth()
```

Out[27]: 21

Ploting the Actual vs Predicted Values for Model 1

In [32]:
```python
# Plot the actual versus predicted values for model 1
plt.scatter(y_test, y_pred1)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--', color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title("Decision Tree Regression - Unspecified Max Depth")
plt.show()
```



b) Setting max depth = 10

In [33]:
```python
# Split the data into predictor variables and the target variable
X = f1_df[['points', 'laps', 'altitude', 'fastestLapDuration', 'averagePitstopDuration', 'averageLapDuration']]
y = f1_df['fastestLapSpeed']

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Fit the model to the training data with max depth = 10
model2 = DecisionTreeRegressor(max_depth = 10, random_state=0)
model2.fit(X_train, y_train)

# Predict the target variable using the test data
y_pred2 = model2.predict(X_test)

# Evaluate the model using metrics such as R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE)
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print('R-squared:', r2_score(y_test, y_pred2))
print('MSE:', mean_squared_error(y_test, y_pred2))
print('MAE:', mean_absolute_error(y_test, y_pred2))
```

```
R-squared: 0.9924327675308171
MSE: 3.4659641803895465
MAE: 1.1119576321075262
```

Ploting the Actual vs Predicted Values for Model 2

In [34]:
```python
# Plot the actual versus predicted values for model 2
plt.scatter(y_test, y_pred2)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--', color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title("Decision Tree Regression - Max Depth = 10")
plt.show()
```

The plots look pretty much simmilar since the model explains 99.24% variability in fastestLapSpeed. It looks like there is overfitting.

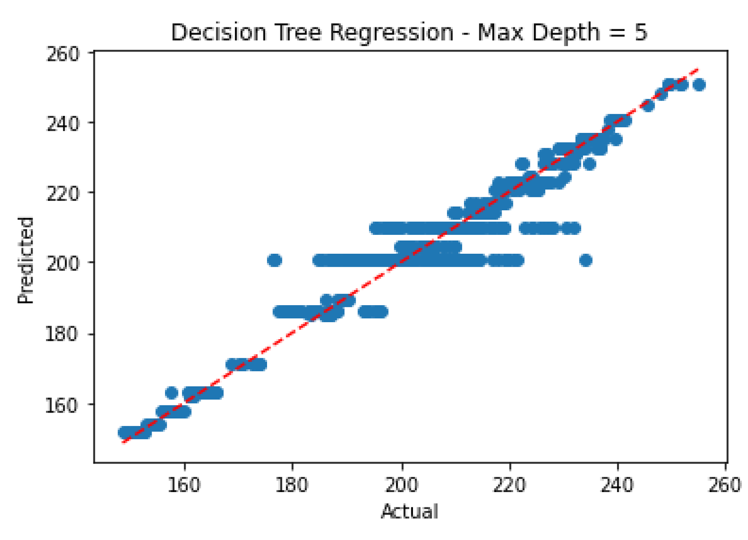
c) Setting max depth = 5

In [35]:
```python
# Split the data into predictor variables and the target variable
X = f1_df[['points', 'laps', 'altitude', 'fastestLapDuration', 'averagePitstopDuration', 'averageLapDuration']]
y = f1_df['fastestLapSpeed']

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Fit the model to the training data with max depth = 5
model3 = DecisionTreeRegressor(max_depth = 5, random_state=0)
model3.fit(X_train, y_train)

# Predict the target variable using the test data
y_pred3 = model3.predict(X_test)

# Evaluate the model using metrics such as R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE)
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print('R-squared:', r2_score(y_test, y_pred3))
print('MSE:', mean_squared_error(y_test, y_pred3))
print('MAE:', mean_absolute_error(y_test, y_pred3))
```

```
R-squared: 0.9165823259049434
MSE: 38.20718758176254
MAE: 4.313254314746929
```

Ploting the Actual vs Predicted Values for Model 3

In [36]:
```python
# Plot the actual versus predicted values for model 3
plt.scatter(y_test, y_pred3)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--', color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title("Decision Tree Regression - Max Depth = 5")
plt.show()
```



When the max depth is set as 5, the model expalains 91.66% variability in fastestLapSpeed. This looks significantly better, and overfitting has been dealt with.

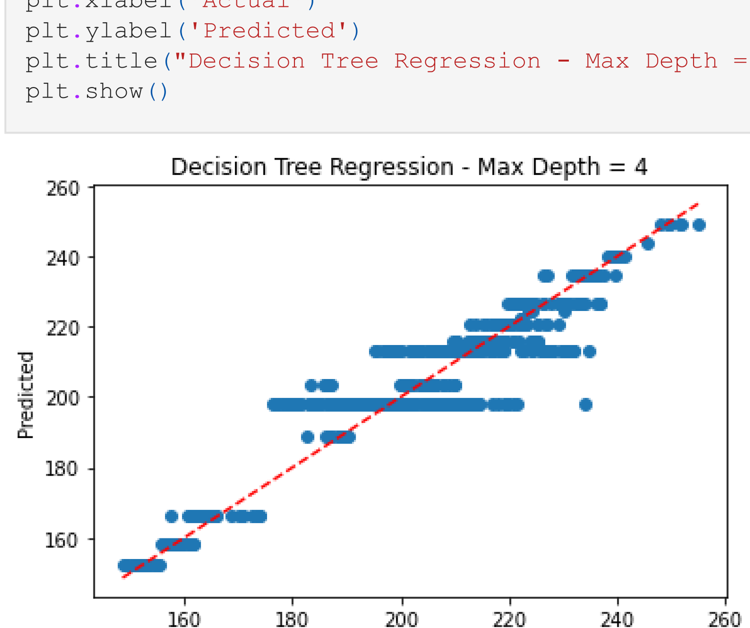
d) Setting max depth = 4

In [37]:
```python
# Split the data into predictor variables and the target variable
X = f1_df[['points', 'laps', 'altitude', 'fastestLapDuration', 'averagePitstopDuration', 'averageLapDuration']]
y = f1_df['fastestLapSpeed']

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Fit the model to the training data with max depth = 4
model4 = DecisionTreeRegressor(max_depth = 4, random_state=0)
model4.fit(X_train, y_train)

# Predict the target variable using the test data
y_pred4 = model4.predict(X_test)

# Evaluate the model using metrics such as R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE)
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print('R-squared:', r2_score(y_test, y_pred4))
print('MSE:', mean_squared_error(y_test, y_pred4))
print('MAE:', mean_absolute_error(y_test, y_pred4))
```

```
R-squared: 0.8678114015812329
MSE: 60.545377592293875
MAE: 5.61180142493019
```

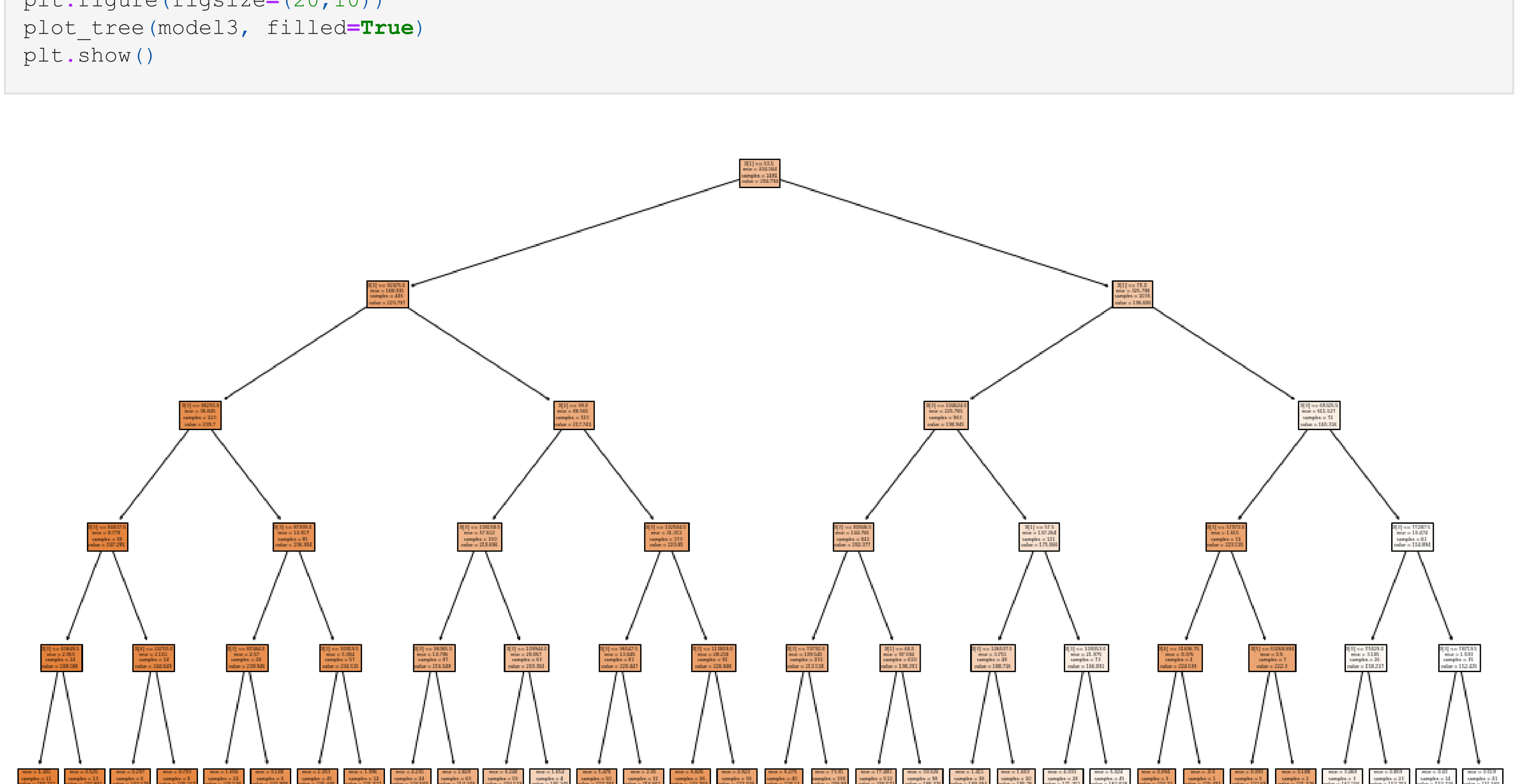
Ploting the Actual vs Predicted Values for Model 4

In [38]:
```python
# Plot the actual versus predicted values for model 4
plt.scatter(y_test, y_pred4)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--', color='red')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title("Decision Tree Regression - Max Depth = 4")
plt.show()
```



When the max depth is set as 4, the model explains 86.78% variability in fastestLapSpeed.

In [44]:
```python
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Plot the decision tree structure
plt.figure(figsize=(20,10))
plot_tree(model3, filled=True)
plt.show()
```



Evaluating the performance of Decision Tree Regression

From the analysis above, it is clear that if we want to obtain the same R squared obtained using Linear Regression, we need to set the depth of our decision regression tree to 5 since this gives us the closest R-Squared value (0.9165823259049434, compared to 0.910815 from the linear model). However, a decision tree regressor may not be an ideal model to use to predict a continous varible when the data is linear. From the multiple regression, the data proved to be linear. Still, this was a nice way to learn how a decision tree regressors work, and when to use/not to use this machine learning method. while a devision tree regressor can be useful when modelling non-linear data, it has its disadvantages. The model is prone to overfitting as seen in this example, and selecting an ideal length of the tree can help to solve this issue. The question is, what is a number that would be considered the appropriate length of the tree?