

Chapter 9. Pushing Boundaries with Ensemble Models

 packtpub.com/mapt/book/big_data_and_business_intelligence/9781784390150/9


Ensemble modeling is a process where two or more models are generated and then their results are combined. In this chapter, we'll cover a random forest, which is a nonparametric modeling technique where multiple decision trees are created during training time, and then the result of these decision trees are averaged to give the required output. It's called a **random forest** because many decision trees are created during training time on randomly selected features.

An analogy of this would be to try to guess the number of pebbles in a glass jar. There are groups of people who try to guess the number of pebbles in the jar. Individually, each person would be very wrong in guessing the number of pebbles in the glass jar, but when you average each of their guesses, the resulting averaged guess would be pretty close to the actual number of pebbles in the jar.

In this chapter, you'll learn how to:

- Work with census data on US earnings and explore this data
- Make decision trees to predict if a person is earning more than \$50K
- Make random forest models and get improved data performance

The census income dataset

 packtpub.com/mapt/book/big_data_and_business_intelligence/9781784390150/9/ch09lvl1sec62/the-census-income-dataset

The following table is a census dataset on income created by the University of California, Irvine:

Columns	Description
<code>age</code>	This refers to the age of a person
<code>work_class</code>	This refers to the type of employment a person is involved in
<code>education</code>	This refers to the education level of a person
<code>marital_status</code>	This refers to whether a person is married or not
<code>occupation</code>	This refers to the type of jobs a person is involved in
<code>relationship</code>	This refers to the type of relationship of the person
<code>race</code>	This refers to the ethnicity of a person
<code>gender</code>	This refers to the gender of a person
<code>hours_per_week</code>	This refers to the average hours worked per week
<code>native_country</code>	This refers to the country of origin
<code>greater_than_50k</code>	This refers to the flag that indicates whether a person is earning more than \$50K in a year

Let's load this data:

```
>>> data = pd.read_csv('./Data/census.csv')
```

Let's check the fill rate of the data:

```
>>> data.count(0)/data.shape[0] * 100
age                100.000000
workclass          94.361179
education          100.000000
education_num      100.000000
marital_status     100.000000
occupation         94.339681
relationship       100.000000
race               100.000000
gender             100.000000
capital_gain       100.000000
capital_loss       100.000000
hours_per_week     100.000000
native_country     98.209459
greater_than_50k   100.000000
dtype: float64
```

We can see that the columns have a good fill rate. We'll remove the rows that have empty values and also remove the `education_num` column as it contains the same information, such as education and its unique codes:

```
>>> data = data.dropna(how='any')
>>> del data['education_num']
```

Exploring the census data

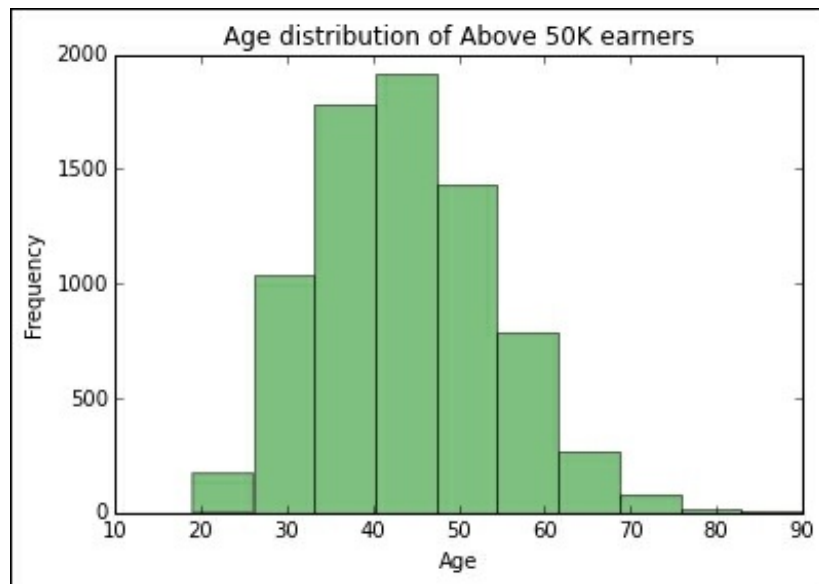
Let's explore the census data and understand the patterns with the data before building the model.

Hypothesis 1: People who are older earn more

We'll create a histogram of people who earn more than \$50K:

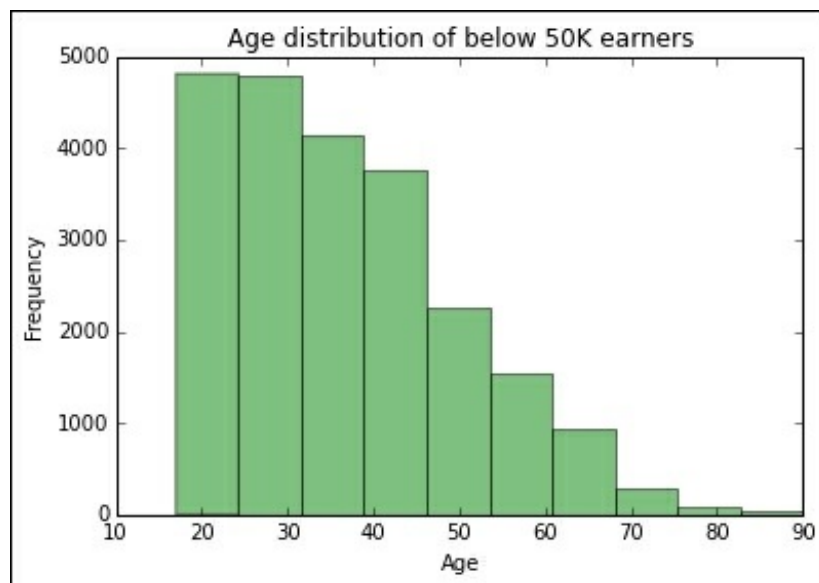
```
>>> hist_above_50 = plt.hist(data[data.greater_than_50k == 1].age.values, 10,
facecolor='green', alpha=0.5)
>>> plt.title('Age distribution of Above 50K earners')
>>> plt.xlabel('Age')
>>> plt.ylabel('Frequency')
```

Here is the histogram for the preceding code:



Now, we'll plot a histogram of the age of the people who earn less than \$50K a year, using this code:

```
>>> hist_below_50 = plt.hist(data[data.greater_than_50k == 0].age.values, 10,
facecolor='green', alpha=0.5)
>>> plt.title('Age distribution of below 50K earners')
>>> plt.xlabel('Age')
>>> plt.ylabel('Frequency')
```

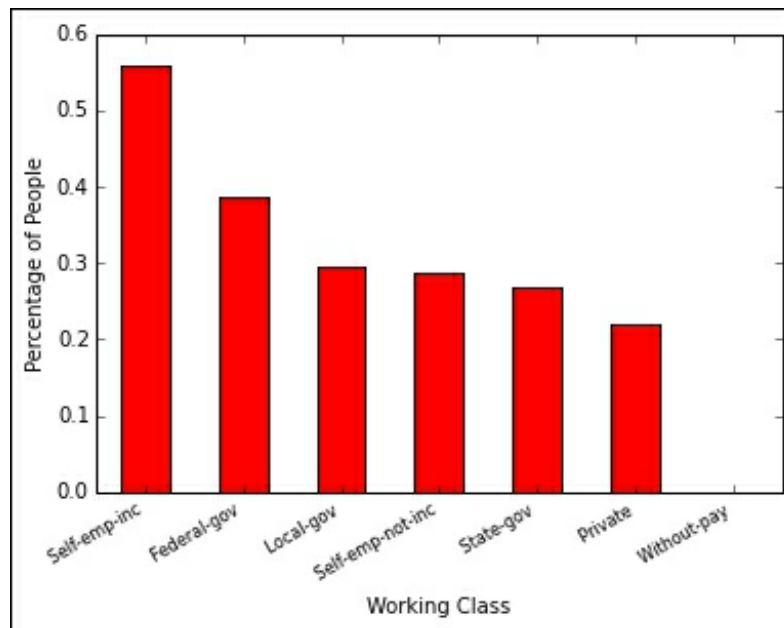


We can see that people who earn above \$50K are mostly aged between their late 30s and mid 50s, whereas people who earn less than \$50K are primarily aged between 20 and 30.

Hypothesis 2: Income bias based on working class

Let's see what the distribution of people earning more than \$50K between different working class groups is. We'll see the percentage of earners who earn more than \$50K in each of the groups, using the following code:

```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('workclass').workclass.count(),
, data[data.greater_than_50k ==
0].groupby('workclass').workclass.count()], axis=1)
>>> dist_data.columns = ['wk_class_gt50', 'wk_class_lt50']
>>> dist_data_final = dist_data.wk_class_gt50 / (dist_data.wk_class_lt50 +
dist_data.wk_class_gt50 )
>>> dist_data_final.sort(ascending=False)
>>> ax = dist_data_final.plot(kind = 'bar', color = 'r', y='Percentage')
>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Working Class')
>>> ax.set_ylabel('Percentage of People')
```

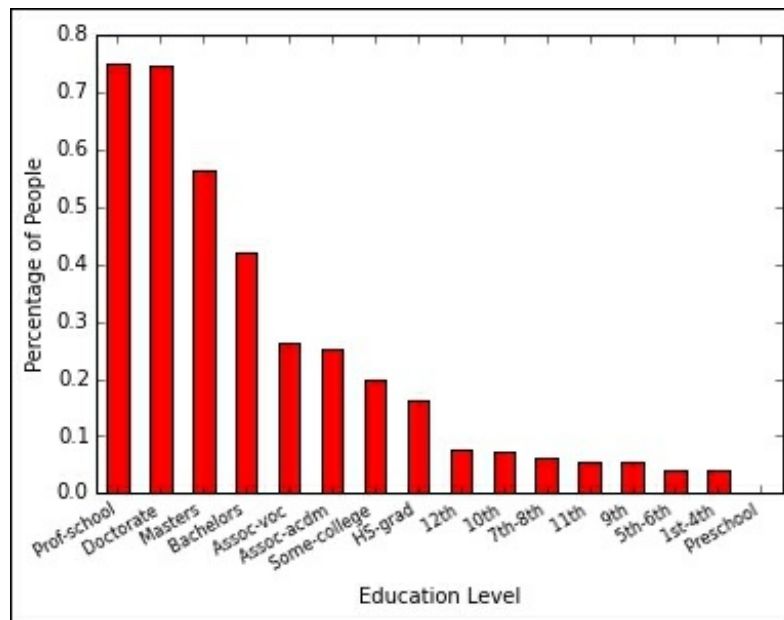


We see that people who are self-employed and have a company have got the maximum share of people who earn more than \$50K. The second most well-off group in terms of earning are federal government employees.

Hypothesis 3: People with more education earn more

Education is an important field. It should be related to the level of earning power of an individual:

```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('education').education.count(),
, data[data.greater_than_50k ==
0].groupby('education').education.count()], axis=1)
>>> dist_data.columns = ['education_gt50', 'education_lt50']
>>> dist_data_final = dist_data.education_gt50 / (dist_data.education_gt50 +
dist_data.education_lt50)
>>> dist_data_final.sort(ascending = False)
>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')
>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Education Level')
>>> ax.set_ylabel('Percentage of People')
```



We can see that the more the person is educated, the greater the number of people in their group who earn more than \$50K.

Hypothesis 4: Married people tend to earn more

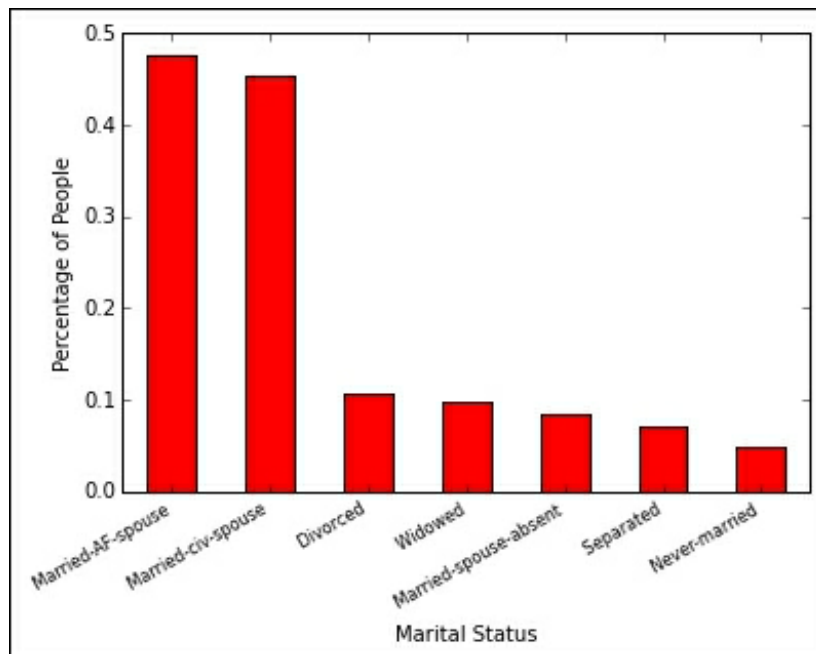
Let's see how distribution is based on marital status:

```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('marital_status').marital_status.count()
, data[data.greater_than_50k ==
0].groupby('marital_status').marital_status.count()], axis=1)
>>> dist_data.columns = ['marital_status_gt50', 'marital_status_lt50']

>>> dist_data_final = dist_data.marital_status_gt50 /
(dist_data.marital_status_gt50+dist_data.marital_status_lt50)

>>> dist_data_final.sort(ascending = False)

>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')
>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Marital Status')
>>> ax.set_ylabel('Percentage of People')
```



We can see that people who are married earn better as compared to people who are single.

Hypothesis 5: There is a bias in income based on race

Let's see how earning power is based on the race of the person:

```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('race').race.count()
, data[data.greater_than_50k == 0].groupby('race').race.count()],
axis=1)

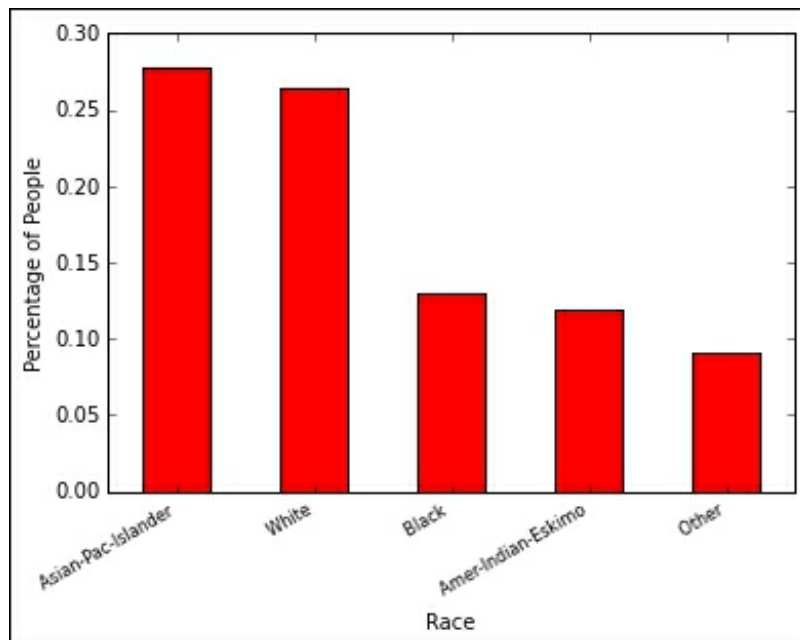
>>> dist_data.columns = ['race_gt50', 'race_lt50']

>>> dist_data_final = dist_data.race_gt50 / (dist_data.race_gt50 +
dist_data.race_lt50 )

>>> dist_data_final.sort(ascending = False)

>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')

>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Race')
>>> ax.set_ylabel('Percentage of People')
```



Asian Pacific people and Whites have the highest earning power.

Hypothesis 6: There is a bias in the income based on occupation

Let's see how earning power is based on the occupation of a person:

```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('occupation').occupation.count()
, data[data.greater_than_50k ==
0].groupby('occupation').occupation.count()], axis=1)

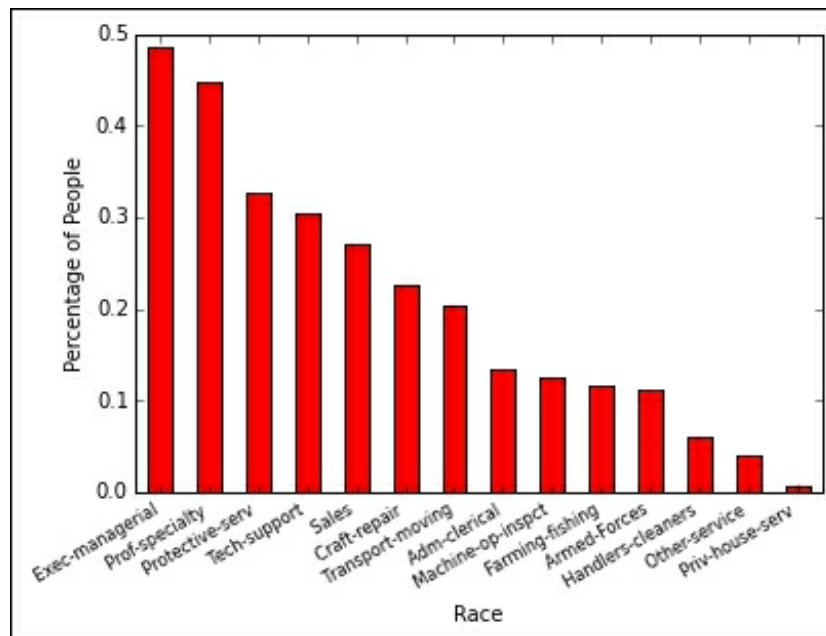
>>> dist_data.columns = ['occupation_gt50', 'occupation_lt50']

>>> dist_data_final = dist_data.occupation_gt50 / (dist_data.occupation_gt50 +
dist_data.occupation_lt50 )

>>> dist_data_final.sort(ascending = False)

>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')

>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Occupation')
>>> ax.set_ylabel('Percentage of People')
```

We can see that people who are in specialized or managerial positions earn more.

Hypothesis 7: Men earn more

Let's see how earning power is based on gender:

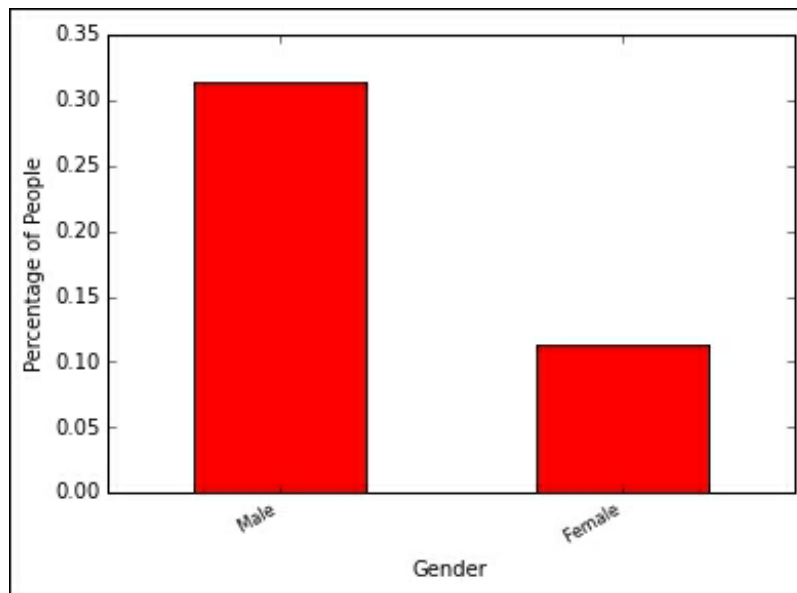
```
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('gender').gender.count(),
                           data[data.greater_than_50k ==
0].groupby('gender').gender.count()], axis=1)
>>> dist_data.columns = ['gender_gt50', 'gender_lt50']

>>> dist_data_final = dist_data.gender_gt50 / (dist_data.gender_gt50 +
dist_data.gender_lt50)

>>> dist_data_final.sort(ascending = False)

>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')

>>> ax.set_xticklabels(dist_data_final.index, rotation=30, fontsize=8, ha='right')
>>> ax.set_xlabel('Gender')
>>> ax.set_ylabel('Percentage of People')
```

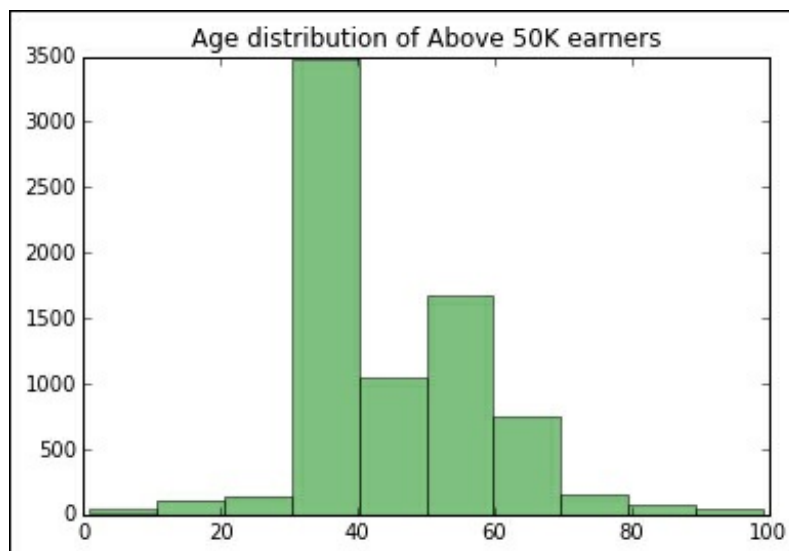


It's no surprise to see that males have a higher earning power as compared to females. It will be good to see the two bars at an equal level sometime in the future.

Hypothesis 8: People who clock in more hours earn more

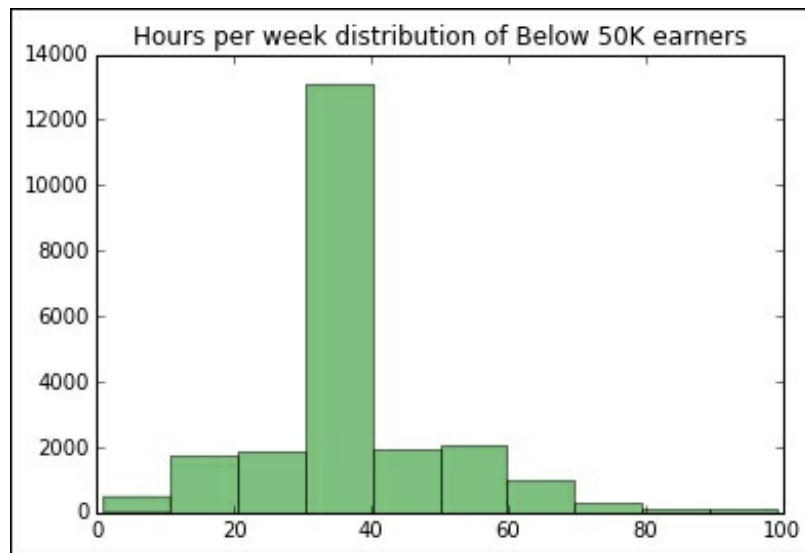
Let's see the distribution of people who earn above \$50K based on their working hours per week:

```
>>> hist_above_50 = plt.hist(data[data.greater_than_50k == 1].hours_per_week.values,  
10, facecolor='green', alpha=0.5)  
>>> plt.title('Hours per week distribution of Above 50K earners')
```



Now, let's see the distribution of the earners below \$50K based on their working hours per week:

```
>>> hist_below_50 = plt.hist(data[data.greater_than_50k == 0].hours_per_week.values,  
10, facecolor='green', alpha=0.5)  
>>> plt.title('Hours per week distribution of Below 50K earners')
```



We can see that people who earn more than \$50K and less than this have an average of 40 working hours per week, but it can be seen that people who earn above \$50K have a higher number of people who work more than 40 hours.

Hypothesis 9: There is a bias in income based on the country of origin

Let's see how earning power is based on a person's country of origin:

```
>>> plt.figure(figsize=(10,5))
>>> dist_data = pd.concat([data[data.greater_than_50k ==
1].groupby('native_country').native_country.count()
, data[data.greater_than_50k ==
0].groupby('native_country').native_country.count()], axis=1)

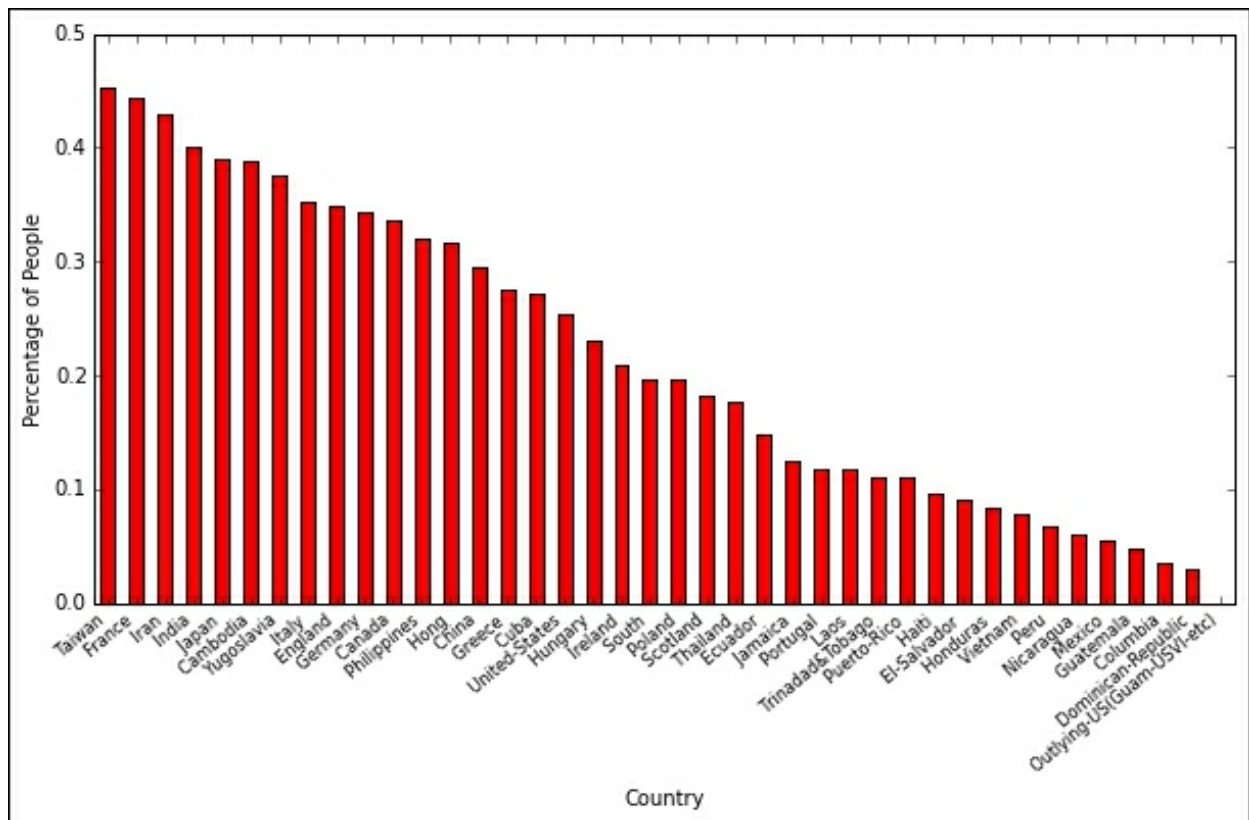
>>> dist_data.columns = ['native_country_gt50', 'native_country_lt50']

>>> dist_data_final = dist_data.native_country_gt50 / (dist_data.native_country_gt50
+ dist_data.native_country_lt50 )

>>> dist_data_final.sort(ascending = False)

>>> ax = dist_data_final.plot(kind = 'bar', color = 'r')

>>> ax.set_xticklabels(dist_data_final.index, rotation=40, fontsize=8, ha='right')
>>> ax.set_xlabel('Country')
>>> ax.set_ylabel('Percentage of People')
```



We can see that Taiwanese, French, Iranians, and Indians are the most well-earning people among different counties.

Decision trees

packtpub.com/mapt/book/big_data_and_business_intelligence/9781784390150/9/ch09lv1sec63/decision-trees

To understand decision tree-based models, let's try to imagine that Google wants to recruit people for a software development job. Based on the employees that they already have and the ones they have rejected previously, we can determine whether an applicant was from an Ivy League college or not and what the **Grade Point Average (GPA)** of the applicant was.

The decision tree will split the applicants into Ivy League and non-Ivy League groups. The Ivy League group will then be split into high GPA and low GPA so that people with a high GPA are likely to be tagged highly and the ones with a low GPA are likely to get recruited.

Applicants who have a high GPA and belong to non-Ivy League colleges have a slightly better chance of getting recruited as compared to those who have a low GPA and belong to non-Ivy League colleges.

The preceding explanation is what a decision tree does in simple terms.

Let's create a decision tree on the basis of our data to predict what the likelihood of a person earning more than \$50K is going to be:

```
>>> data_test = pd.read_csv('./Data/census_test.csv')
>>> data_test = data_test.dropna(how='any')
>>> formula = 'greater_than_50k ~ age + workclass + education + marital_status +
occupation + race + gender + hours_per_week + native_country '
>>> y_train,x_train = dmatrices(formula, data=data, return_type='dataframe')
>>> y_test,x_test = dmatrices(formula, data=data_test, return_type='dataframe')
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(x_train, y_train)
```

Let's see how the model performs:

```
>>> from sklearn.metrics import classification_report
>>> y_pred = clf.predict(x_test)
>>> print pd.crosstab(y_test.greater_than_50k
                      ,y_pred
                      ,rownames = ['Actual']
                      ,colnames = ['Predicted'])

>>> print '\n \n'
>>> print classification_report(y_test.greater_than_50k,y_pred)
```

Predicted	0	1			
Actual					
0	9832	1528			
1	1781	1919			
			precision	recall	f1-score
	0.0	0.85	0.87	0.86	11360
	1.0	0.56	0.52	0.54	3700
avg / total		0.78	0.78	0.78	15060

We can see that the people who don't earn more than \$50K can be predicted well as there is a precision of 85% and a recall of 87%. People who earn more than \$50K can only be predicted with a precision of 56% and a recall of 52%

Note that the order of the dependent variables given in the formula will change these values slightly. You can experiment to see whether changing the order of the variables will improve their precision/recall.

Random forests

packtpub.com/mapt/book/big_data_and_business_intelligence/9781784390150/9/ch09lv1sec64/random-forests

We have learned how to create a decision tree but, at times, decision tree models don't hold up well when there are many variables and a large dataset. This is where ensemble models, such as random forest, come to rescue.

A random forest basically creates many decision trees on the dataset and then averages out the results. If you see a singing competition, such as American Idol, or a sporting competition, such as the Olympics, there are multiple judges. The reason for having multiple judges is to eliminate bias and give fair results, and this is what a random forest tries to achieve.

A decision tree can change drastically if the data changes slightly and it can easily overfit the data.

Let's try to create a random forest model and see how its precision/recall is compared to the decision tree that we just created:

```
>>> import sklearn.ensemble as sk
>>> clf = sk.RandomForestClassifier(n_estimators=100)
>>> clf = clf.fit(x_train, y_train.greater_than_50k)
```

After building the model, let's cross-validate the model on the test data:

```
>>> y_pred = clf.predict(x_test)
>>> print pd.crosstab(y_test.greater_than_50k
                      , y_pred
                      , rownames = ['Actual']
                      , colnames = ['Predicted'])
>>> print '\n \n'
>>> print classification_report(y_test.greater_than_50k, y_pred)
```

Predicted	0	1
Actual		
0	10148	1212
1	1685	2015

	precision	recall	f1-score	support
0.0	0.86	0.89	0.88	11360
1.0	0.62	0.54	0.58	3700
avg / total	0.80	0.81	0.80	15060

We can see that we have improved the precision and recall for the people who don't earn more than \$50K, as well as for the people who do.

Let's try to do some fine-tuning to achieve better performance for the model by using the `min_samples_split` parameter and setting it to `5`. This parameter tells us that the minimum number of samples required to create a split is `5`:

```
>>> clf = sk.RandomForestClassifier(n_estimators=100,
oob_score=True,min_samples_split=5)
>>> clf = clf.fit(x_train, y_train.greater_than_50k)
>>> y_pred = clf.predict(x_test)
>>> print pd.crosstab(y_test.greater_than_50k
                        ,y_pred
                        ,rownames = ['Actual']
                        ,colnames = ['Predicted'])

>>> print '\n \n'
>>> print classification_report(y_test.greater_than_50k,y_pred)
```

Predicted	0	1
Actual		
0	10269	1091
1	1636	2064

	precision	recall	f1-score	support
0.0	0.86	0.90	0.88	11360
1.0	0.65	0.56	0.60	3700
avg / total	0.81	0.82	0.81	15060

We increased the recall of 0% to 90%, 1% to 56%, and the precision of 1% to 65%.

We'll fine-tune the model further by increasing the minimum number of leaves to `2` by using the `min_leaf` parameter. The meaning of this parameter indicates that the minimum number of nodes to be created are `2`:

```
>>> clf = sk.RandomForestClassifier(n_estimators=100,
oob_score=True,min_samples_split=5, min_samples_leaf= 2)
>>> clf = clf.fit(x_train, y_train.greater_than_50k)

>>> y_pred = clf.predict(x_test)

>>> print pd.crosstab(y_test.greater_than_50k
                        ,y_pred
                        ,rownames = ['Actual']
                        ,colnames = ['Predicted'])

>>> print '\n \n'

>>> print classification_report(y_test.greater_than_50k,y_pred)
```

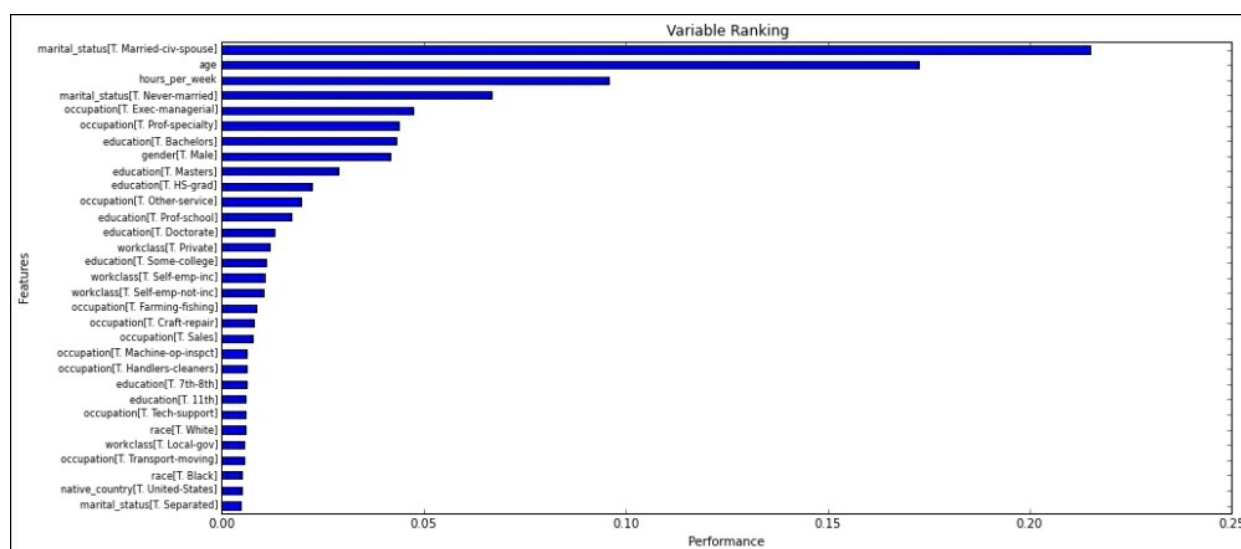

Predicted	0	1
Actual		
0	10453	907
1	1621	2079

	precision	recall	f1-score	support
0.0	0.87	0.92	0.89	11360
1.0	0.70	0.56	0.62	3700
avg / total	0.82	0.83	0.83	15060

We have further significantly increased the recall of 0% to 92% and the precision of 1% to 70%. This model performs decently.

Let's see the importance of the variables that are contributing to the prediction. We'll use the feature importance attribute of the `clf` object, and using this, we'll plot important features, such as dependent variables that are sorted by their importance:

```
>>> model_ranks = pd.Series(clf.feature_importances_, index=x_train.columns,
name='Importance').sort(ascending=False, inplace=False)
>>> model_ranks.index.name = 'Features'
>>> top_features = model_ranks.iloc[:31].sort(ascending=True, inplace=False)
>>> plt.figure(figsize=(15,7))
>>> ax = top_features.plot(kind='barh')
>>> _ = ax.set_title("Variable Ranking")
>>> _ = ax.set_xlabel('Performance')
>>> _ = ax.set_yticklabels(top_features.index, fontsize=8)
```



We can see that those people who are married to a civilian spouse are very good indicators of whether a particular group of people earn more than \$50K or not. This is followed by the age of a person, and finally, the number of hours a week a person works. Also, people who aren't married are good indicators of predicting the group of people who earn less than \$50K.

Summary



packtpub.com/mapt/book/big_data_and_business_intelligence/9781784390150/9/ch09lvl1sec65/summary

In this chapter, we explored the patterns in the census data and then understood how a decision tree was constructed and also built a decision tree model on the data given. You then learned the concept of ensemble models with the help of a random forest and improved the performance of prediction by using the random forest model.

In the next chapter, you'll learn clustering, which is basically grouping elements together that are similar to each other. We will use the k-means cluster for this.