Problem 0 & 1.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [39]:
```
# read in data
train = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience - HW5
trainX = train.loc[:,'F1':]
trainY = train['Y']
testX = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience - HW5
```

In [40]:
```
realData = pd.read_csv('/Users/aparnaaidith/Desktop/Business DataScience -
```

1) Model interpretability

What is the effect of MonthlyIncome to the prediction? Quantify as much as you can how 1000, 2000 or 3000 extra per month affect the probablity of deliquency. Do this by fitting a simple model on the dataset and using your best model.

In [41]:
```
from sklearn.linear_model import ElasticNetCV

import xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
```

In [42]:
```
realX = realData[realData.columns[1:]]
realY = realData[realData.columns[0]]
```

In [43]:
```
from sklearn.model_selection import cross_val_score

xgb_BOOSTER = XGBClassifier(learning_rate=0.1, n_estimators= 200, max_depth
scores = cross_val_score(xgb_BOOSTER, realX, realY, cv=5, scoring = 'roc_au
print("Avg Score: {}% ({})".format(scores.mean()*100, scores.std()*100))
```

```
Avg Score: 86.37090968418015% (0.2984484605833362)
```

In [44]:
```
alteredX = realX.copy()

xgb.fit(realX,realY)

real_pred = xgb.predict_proba(realX)[:,1]
for i in range(1000,4000,1000):
    alteredX['MonthlyIncome'] = realX['MonthlyIncome'] + i
    preds = xgb.predict_proba(alteredX)[:,1]
    print('Percent decrease in avg delinquincy rate with ${} in monthly inc
        -100*(preds.mean()-real_pred.mean())/real_pred.mean()))
```

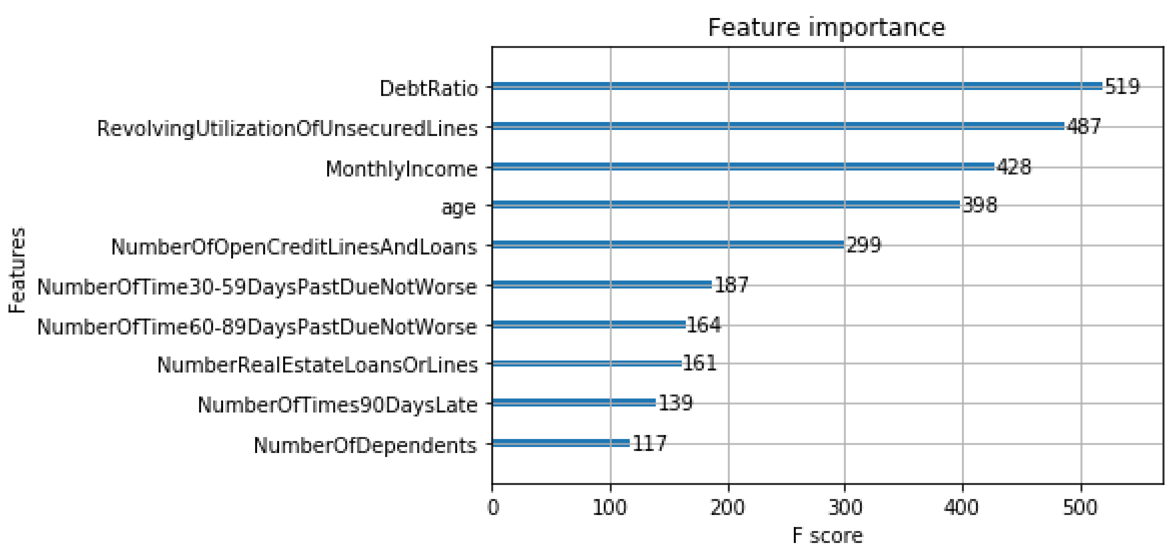
```
Percent decrease in avg delinquincy rate with $1000 in monthly income: 0.
9590820261309075%
Percent decrease in avg delinquincy rate with $2000 in monthly income: 2.
5288837980835255%
Percent decrease in avg delinquincy rate with $3000 in monthly income: 3.
9909223277206913%
```

With no other changes, increasing the monthly income of a sample will decrease the model predicting delinquincy. However, this does not account for how other features may change as a result of the change in income.

2) What is the most important variable in predicting deliquency? What is the most important pair of variables? Make a data science argument supported by data.

In [45]:
```
xgboost.plot_importance(xgb)
# xgb.feature_importances_
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x10f755a58>



In [46]:
```
from sklearn.feature_selection import SelectKBest,chi2
realX = realX.fillna(realX.mean())
for i in range(1,3):
    b = SelectKBest(chi2,i)
    b.fit(realX,realY)
    print('best {} features - chi score: {}'.format(i,realX.columns.get_val
    print('best {} features - roc_auc xgboost: {}'.format(i,realX.columns.g
```

```
best 1 features - chi score: ['MonthlyIncome']
best 1 features - roc_auc xgboost: ['DebtRatio']
best 2 features - chi score: ['NumberOfTimes90DaysLate' 'MonthlyIncome']
best 2 features - roc_auc xgboost: ['RevolvingUtilizationOfUnsecuredLine
s' 'DebtRatio']
```

Depending on the scoring function and the model, the best features are different, as listed above. According to the chi scorer, the best pair of features are MonthlyIncome and NumberOfTimes90DaysLate and the best single feature is MonthlyIncome

3) The Age Discrimination in Employment Act (ADEA) forbids age discrimination against people who are age 40 or older. Look at the best models you used in your Kaggle competion. Were they discriminating against older people? Make the best argument you can.

In [47]:
```
X = pd.concat([trainX,testX])
X1 = X[[a for a in X.columns if a != 'F5' and a != 'F19']]
Y1 = X['F5'][X['F5'].notnull()]
Y2 = X['F19'][X['F19'].notnull()]
f5train = X1[X['F5'].notnull()]
f19train = X1[X['F19'].notnull()]
f5test = X1[X['F5'].isnull()]
f19test = X1[X['F19'].isnull()]
enet = ElasticNetCV(alphas=[0.1,1,5,10],l1_ratio=[0.01,0.1,0.3,0.5,0.7,0.9]
enet.fit(f5train,Y1)
yf5 = enet.predict(f5test)
enet = ElasticNetCV(alphas=[0.1,1,5,10],l1_ratio=[0.01,0.1,0.3,0.5,0.7,0.9]
enet.fit(f19train,Y2)
yf19 = enet.predict(f19test)
```

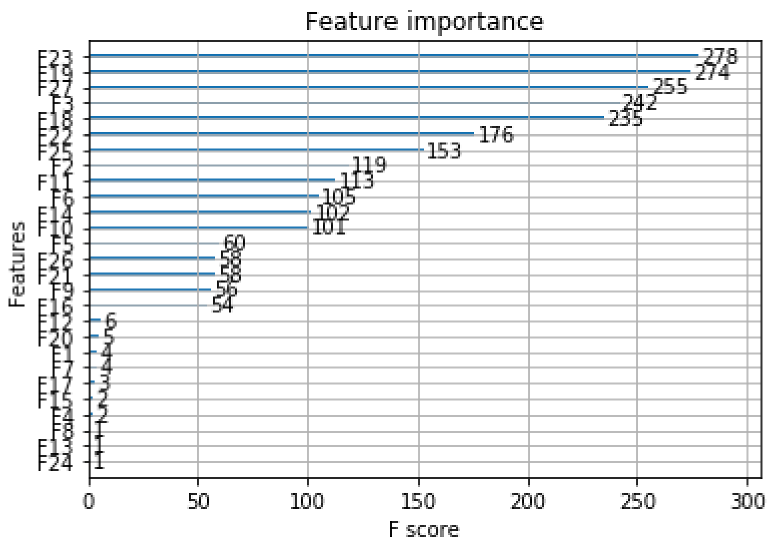
```
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/model_s
election/_split.py:1943: FutureWarning: You should specify a value for 'c
v' instead of relying on the default value. The default value will change
from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
  ConvergenceWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
  ConvergenceWarning)
/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/coordinate_descent.py:491: ConvergenceWarning: Objective did not co
nverge. You might want to increase the number of iterations. Fitting data
with very small alpha may cause precision problems.
```

In [48]:
```
Yt5 = X['F5'][X['F5'].isnull()]
df = pd.Series(yf5,index=Yt5.index)
X['F5'] = X['F5'].fillna(df)
Yt19 = X['F19'][X['F19'].isnull()]
df = pd.Series(yf19,index=Yt19.index)
X['F19'] = X['F19'].fillna(df)
```

In [49]:
```
trainX = X[:49998]
testX = X[49998:]
```

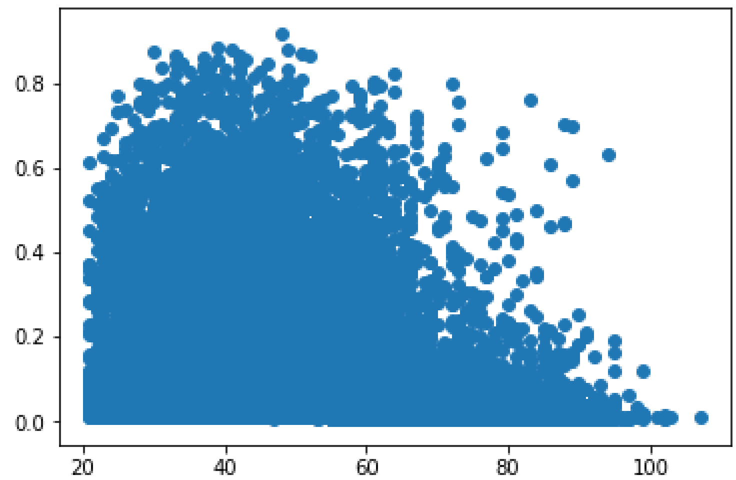
In [50]:
```
) = XGBClassifier(learning_rate=0.1, n_estimators= 200, max_depth= 6, min_ch
:am = xgb.get_params()
)train = xgboost.DMatrix(trainX.values,label=trainY.values)
 = xgboost.cv(param,xgbtrain,num_boost_round=param['n_estimators'],nfold=5,m
).set_params(n_estimators=cv.shape[0])
).fit(trainX,trainY,eval_metric='auc')
)oost.plot_importance(xgb)
```

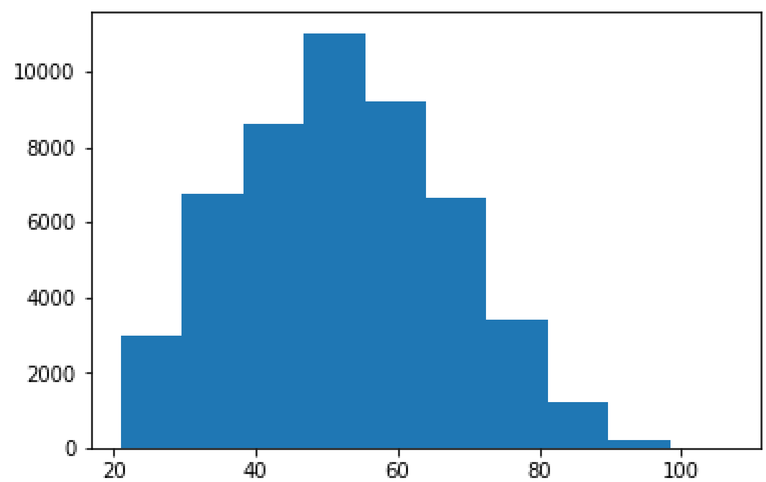
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x111c2e320>



Using our best submission, it seems as if age (F26) is not as important as any of other features (excluding number of dependents F5).

In [51]:
```
plt.scatter(trainX['F26'].values,xgb.predict_proba(trainX)[:,1])
plt.show()
plt.hist(trainX['F26'])
plt.show()
```





We can see that although most there are more predictions leaning towards delinquency in ages under 40, there are more people at low ages. The number of people over 40 are more sparse, but we see relatively the same distribution of predictions. Furthermore, it seems that people over 60 are less likely to be predicted as delinquents

4) Your manager asks if the number of dependents in the family (spouse, no of children) has an effect on loan deliquency. What does the data say? Calculate a p-value to express how confident you are.

In [52]:
```
realX = realData[realData.columns[1:]]
realY = realData[realData.columns[0]]
```

In [53]:
```
realData.groupby('NumberOfDependents')['SeriousDlqin2yrs'].mean()
```

Out[53]:
```
NumberOfDependents
0.0     0.058629
1.0     0.073529
2.0     0.081139
3.0     0.088263
4.0     0.103774
5.0     0.091153
6.0     0.151899
7.0     0.098039
8.0     0.083333
9.0     0.000000
10.0    0.000000
13.0    0.000000
20.0    0.000000
Name: SeriousDlqin2yrs, dtype: float64
```

In [54]:
```
from sklearn.feature_selection import SelectKBest,chi2
realX = realX.fillna(realX.mean())
b = SelectKBest(chi2,10)
b.fit(realX,realY)
print('p-value for effect of number of dependents on delinquency:',b.pvalue
```

```
p-value for effect of number of dependents on delinquency: 1.398057514620
496e-110
```

According to the data, the chance of delinquency increases as the number of dependents approaches 6, then decreases again. The p-value for the number of dependents is negligibly small, so we can safely reject the null hypothesis.

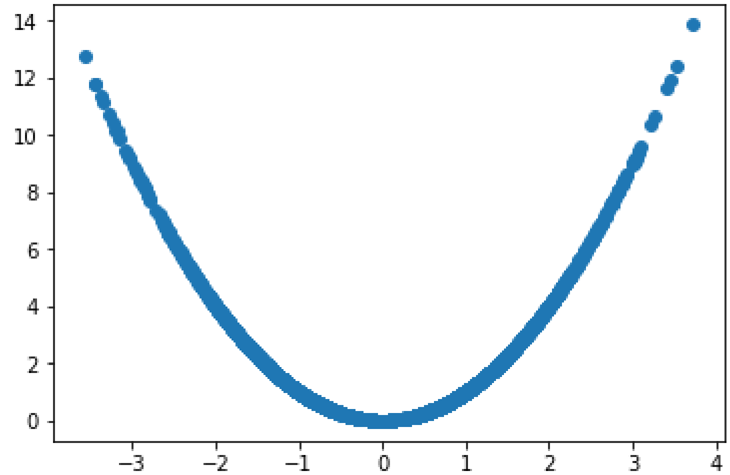
Problem 2.

b)

Create two continuous random variables X, Y so that X and Y are strongly dependent but the best linear regression fit y = β1x+β0 has the optimal β1 = 0. Show a scatter plot ofx, ypairs.

For the problem, we can use the example: in this case X is the a normal distrbution with E(X) = 0 and a Var(X) = 1, and have Y = X^2. These are uncorrelated variables but are also clearly depedent, as shown from the proof from written

In [3]:
```
n=10000
X = np.random.normal(size=n)
Y = X**2
plt.scatter(X,Y)
plt.show()
```



In [4]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(np.reshape(X,(n,1)),Y)
print('B1: ',linreg.coef_[0])
```

```
B1:  0.018186748359344623

/Users/aparnaaidith/py_37_env/lib/python3.7/site-packages/sklearn/linear_
model/base.py:485: RuntimeWarning: internal gelsd driver lwork query erro
r, required iwork dimension not returned. This is likely the result of LA
PACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling ba
ck to 'gelss' driver.
  linalg.lstsq(X, y)
```

Thus, the with the equation y = β1 * x + β0, β1 is close to 0 as β1 represent the correlation between the two variables, thus creating the ideal linear regression.

In [ ]:

(a) Let X be a bernoulli random variable with $p = 1/2$.

Let Y be a random variable that is -1 with probability $\frac{1}{2}$ and 1 with probability $\frac{1}{2}$.

Let $Z = XY$
X and Y are uncorrelated but dependent, shown by proof below:

PROOF

$$\mathbb{E}[Z] = \mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y].$$

Since $\mathbb{E}[Y] = -1 * \frac{1}{2} + 1 * \frac{1}{2} = \underline{\underline{0}}$.

Then $\mathbb{E}[Z] = 0$

$$\mathbb{E}[X] = p = \frac{1}{2}.$$

$$\begin{aligned}
\operatorname{cov}(Z, X) &= \mathbb{E}\left[\left(Z - \mathbb{E}(Z)\right)\left(X - \mathbb{E}(X)\right)\right] \\
&= \mathbb{E}\left[(XY - 0)\left(X - \tfrac{1}{2}\right)\right] \\
&= \mathbb{E}\left[X^2Y - \tfrac{1}{2}XY\right] \\
&= \mathbb{E}\left[Y\left(X^2 - \tfrac{1}{2}X\right)\right] \\
&= \mathbb{E}[Y]\,\mathbb{E}\left[X^2 - \tfrac{1}{2}X\right] = \underline{\underline{0}}.
\end{aligned}$$

Scanned with CamScanner

This shows X and Z are uncorrelated.

For X and Z to be independent, the following must be true:

$$P(Z = z \mid X = x) = P(Z = z)$$

However, this fails if $z = 1$ and $x = 0$.

$\therefore P(Z = 1 \mid X = 0) = 0$, while $P(Z = 1) = \frac{1}{4}$.

Thus, X and Z are independent.

(b) Let x and y be 2 continous random variable.

Let $y = x^2$.

To prove: Find a best fit linear regression $y = \beta_1 x + \beta_0$ which has as optimal $\hat{y} = \beta_1 x + \beta_0$.

Squared loss error $= (y - \hat{y})^2$

$= (x^2 - \beta_1 x - \beta_0)^2$.

$$\int_{-1}^{1} (x^2 - \beta_1 x - \beta_0)^2 \cdot dx$$

$$= \int_{-1}^{1} x^4 + \beta_0^2 + \beta_1^2 x^2 - 2x^2\beta_0 - 2x^3\beta_1 + 2\beta_0\beta_1 x \, dx$$

$$= \frac{x^5}{5} - \frac{2x^3\beta_0}{3} - \frac{2x^4\beta_1}{4} + 2\frac{\beta_0\beta_1 x^2}{2} + \beta_0^2 x + \frac{\beta_1^2 x^3}{3}$$

Substituting 1 & -1:

$$\left(\frac{1}{5} - \frac{2\beta_0}{3} - \frac{2\beta_1}{4} + \frac{2\beta_0\beta_1}{2} + \beta_0^2 + \frac{\beta_1^2}{3}\right) -$$

$$\left(\frac{-1}{5} + \frac{2\beta_0}{3} - \frac{2\beta_1}{4} + 2\frac{\beta_0\beta_1}{2} - \beta_0^2 - \frac{\beta_1^2}{3}\right)$$

$$= \left(\frac{1}{5} + \beta_0^2 + \frac{\beta_1^2}{3} - \frac{2\beta_0}{3} - \frac{\cancel{2\beta_1}}{\cancel{4}} + \frac{\cancel{2\beta_0\beta_1}}{\cancel{2}} \right.$$

$$\left. + \frac{1}{5} + \beta_0^2 + \frac{\beta_1^2}{3} - \frac{2\beta_0}{3} + \frac{\cancel{2\beta_1}}{\cancel{4}} - \frac{\cancel{2\beta_0\beta_1}}{\cancel{2}} \right)$$

$$= \frac{2}{5} + 2\beta_0^2 + \frac{2\beta_1^2}{3} - \frac{4\beta_0}{3}$$

Partial derivatives :-

(1) $$\frac{\partial}{\partial\beta_0}\left(\frac{2}{5} + 2\beta_0^2 + \frac{2\beta_1^2}{3} - \frac{4\beta_0}{3}\right) \Rightarrow 0 + 4\beta_0 + 0 - \frac{4}{3} = 0.$$

$$4\beta_0 = \frac{4}{3}$$

$$\beta_0 = \frac{4}{3} \times \frac{1}{4} = \frac{1}{3}.$$

(2) $$\frac{\partial}{\partial\beta_1}\left(\frac{2}{5} + 2\beta_0^2 + \frac{2\beta_1^2}{3} - \frac{4\beta_0}{3}\right) \Rightarrow 0 + 0 + \frac{4\beta_1}{3} - 0 = 0$$

$$\frac{4\beta_1}{3} = 0.$$

$$\beta_1 = 0.$$

Problem 3.

a) Install Tensorflow and Keras. Complete this tutorial:

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
32768/29515 [==================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
8192/5148 [==============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
4423680/4422102 [==============================] - 0s 0us/step
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
train_images.shape
```

```
(60000, 28, 28)
```

```
len(train_labels)
```

```
60000
```

```
train_labels
```

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

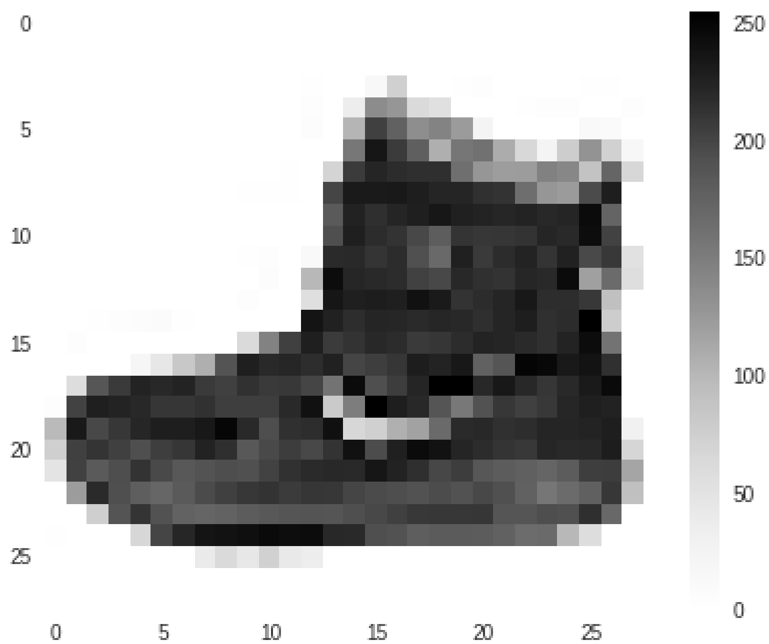
```
test_images.shape
```

```
(10000, 28, 28)
```

```
len(test_labels)
```

```
10000
```

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
```



```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
```


```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.4936 - ac
Epoch 2/5
60000/60000 [==============================] - 5s 76us/step - loss: 0.3730 - ac
Epoch 3/5
60000/60000 [==============================] - 5s 75us/step - loss: 0.3342 - ac
Epoch 4/5
60000/60000 [==============================] - 4s 75us/step - loss: 0.3101 - ac
Epoch 5/5
60000/60000 [==============================] - 4s 75us/step - loss: 0.2957 - ac
<tensorflow.python.keras.callbacks.History at 0x7f0bc03ff5f8>
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 0s 38us/step
Test accuracy: 0.875
```

```
predictions = model.predict(test_images)
```

```
predictions[0]
```

```
    array([2.7209355e-06, 2.3998481e-08, 2.1429447e-07, 1.2474297e-09,
```

```
np.argmax(predictions[0])
```

```
9
```

```
test_labels[0]
```

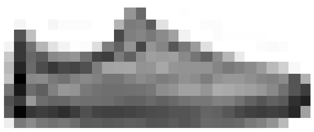
```
9
```

```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

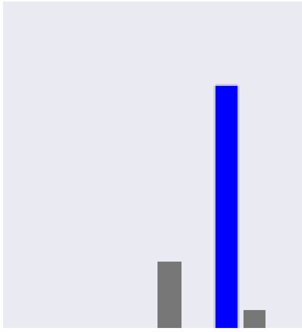
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
```

CODE TEXT

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
```



Sneaker 74% (Sneaker)



```
# Plot the first X test images, their predicted label, and the true label
# Color correct predictions in blue, incorrect predictions in red
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions, test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions, test_labels)
```

```
# Grab an image from the test dataset
img = test_images[0]

print(img.shape)
```

```
(28, 28)
```

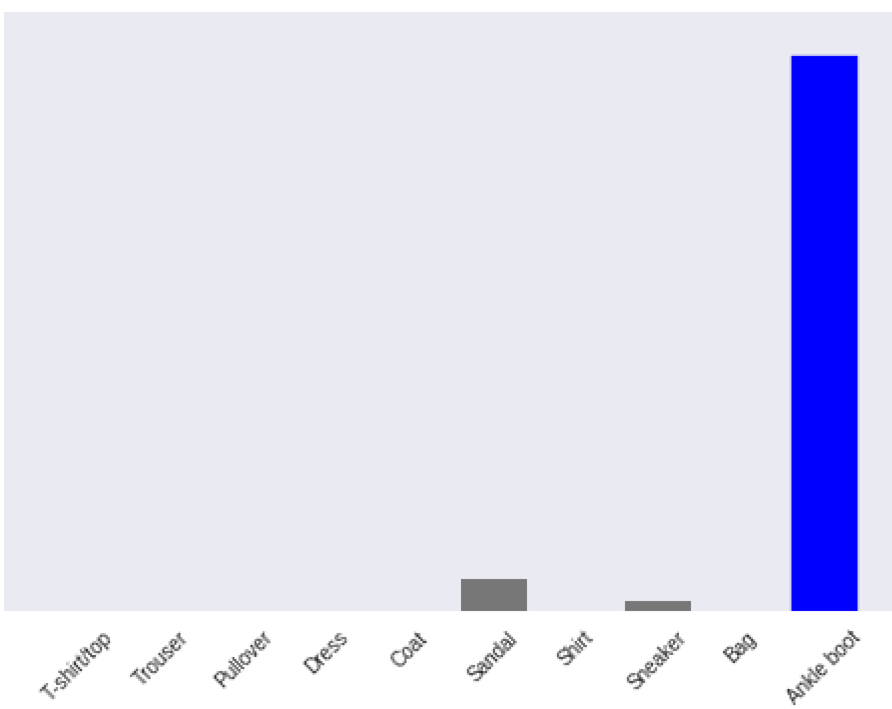
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

```
(1, 28, 28)
```

```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[2.7209351e-06 2.3998478e-08 2.1429403e-07 1.2474296e-09 1.6836999e-07
  5.4479115e-02 1.6102676e-06 1.7795969e-02 4.6110285e-06 9.2771554e-01]]
```

```
plot_value_array(0, predictions_single, test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```

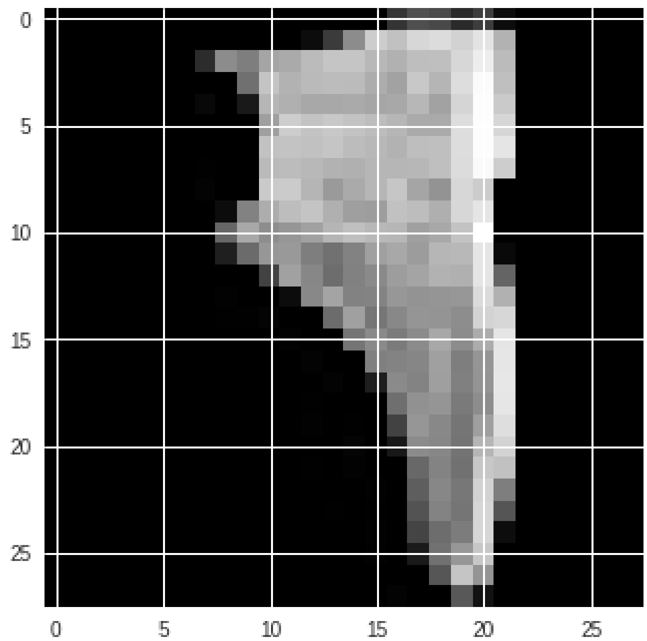



```
np.argmax(predictions_single[0])
```

b) Create 5 new images by modifying current ones (e.g. by rotation, translation or stretching). Try your best model on them. How accurate is it? Should you be modifying images from the training set or test set?

```
image = test_images[0]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bced4f518>
```

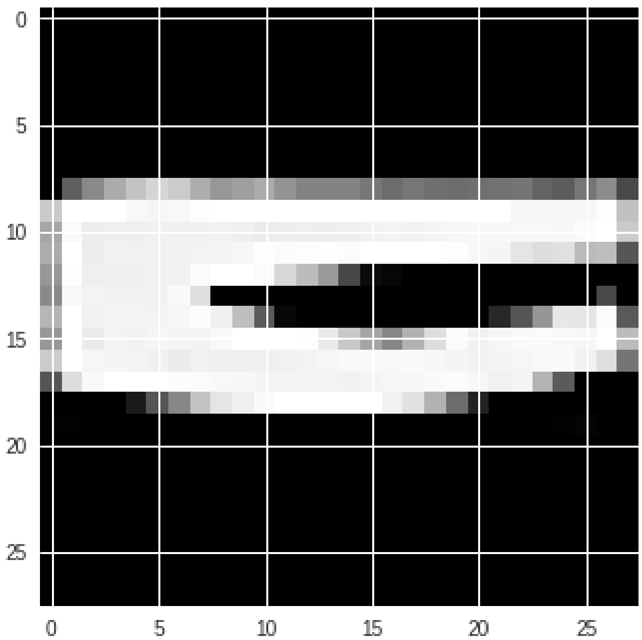



```
image = test_images[1]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

<matplotlib.image.AxesImage at 0x7f0bc03c53c8>

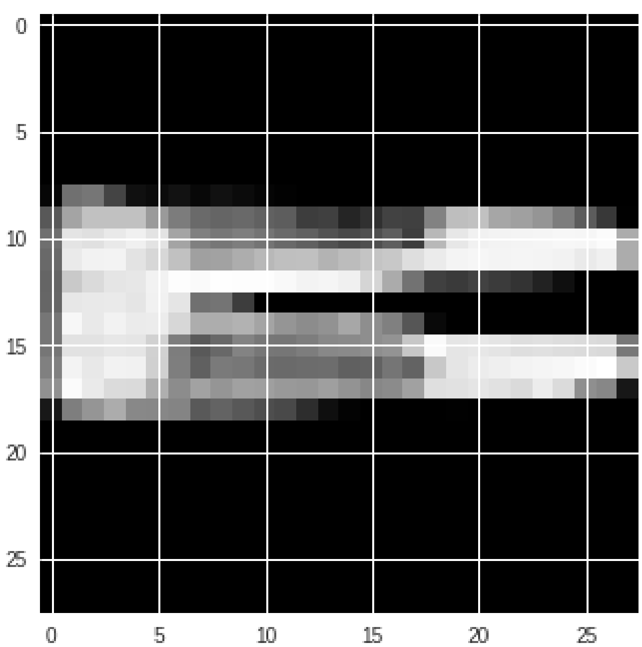
```
image = test_images[2]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

<matplotlib.image.AxesImage at 0x7f0bbd52cd68>



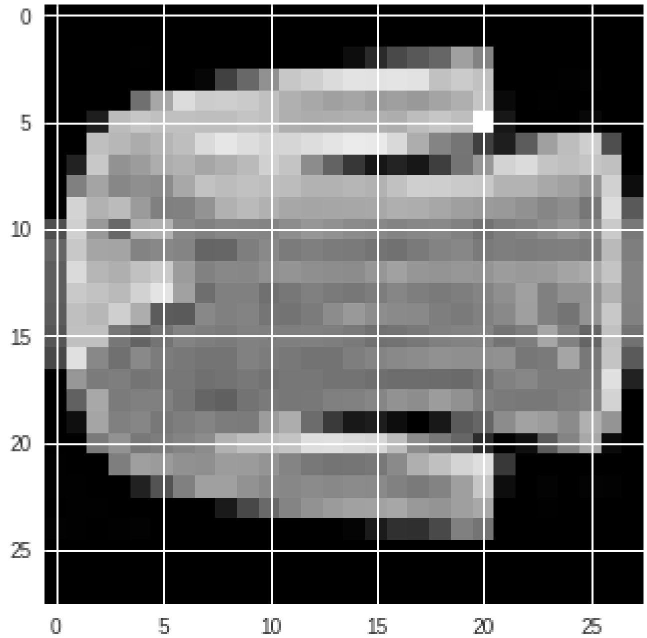

```
image = test_images[3]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

<matplotlib.image.AxesImage at 0x7f0bb9506b00>




```
image = test_images[4]
image_2 = np.rot90(image)
plt.imshow(image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bbff165f8>
```




```python
# Add the image to a batch where it's the only member.
new_img = (np.expand_dims(image_2,0))

print(new_img.shape)
```

```
(1, 28, 28)
```

```python
new_predictions_single = model.predict(new_img)

print(new_predictions_single)
```

```
[[6.4677835e-02 2.1164182e-04 1.0499120e-02 3.7047594e-05 1.5458831e-03
  3.3092888e-06 7.2778083e-02 8.2773018e-05 8.5013485e-01 2.9419767e-05]]
```

```python
new_predictions_single[0]
```

```
array([6.4677835e-02, 2.1164182e-04, 1.0499120e-02, 3.7047594e-05,
       1.5458831e-03, 3.3092888e-06, 7.2778083e-02, 8.2773018e-05,
       8.5013485e-01, 2.9419767e-05], dtype=float32)
```

```python
np.argmax(new_predictions_single[0])
```

```
8
```

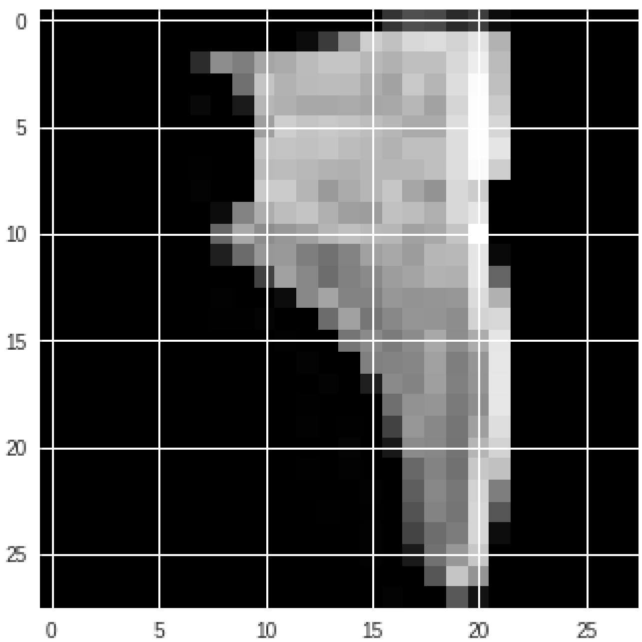
```python
test_labels[0]
```

```
9
```

```python
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_predictions_single, test_labels, new_img)
plt.subplot(1,2,2)
plot_value_array(i, new_predictions_single,  test_labels)
```


```
boot_image = test_images[0]
boot_image_2 = np.rot90(boot_image)
plt.imshow(boot_image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bbd515208>
```




```
# Add the image to a batch where it's the only member.
new_boot_img = (np.expand_dims(boot_image_2,0))

print(new_boot_img.shape)
```

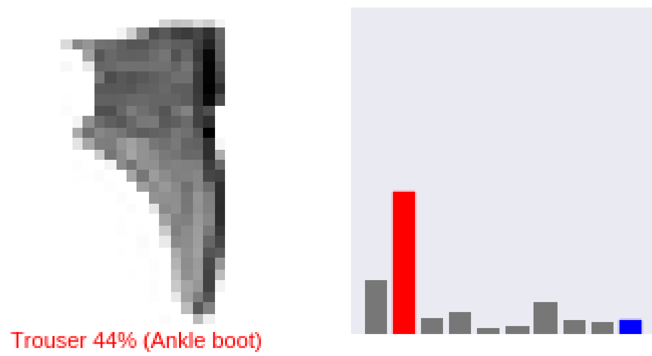
```
(1, 28, 28)
```

```
new_boot_predictions_single = model.predict(new_boot_img)

print(new_boot_predictions_single)
```

```
[[0.1673985  0.4369683  0.05170421 0.0705625  0.01976415 0.0258841
  0.10230368 0.04151233 0.04052156 0.04338068]]
```

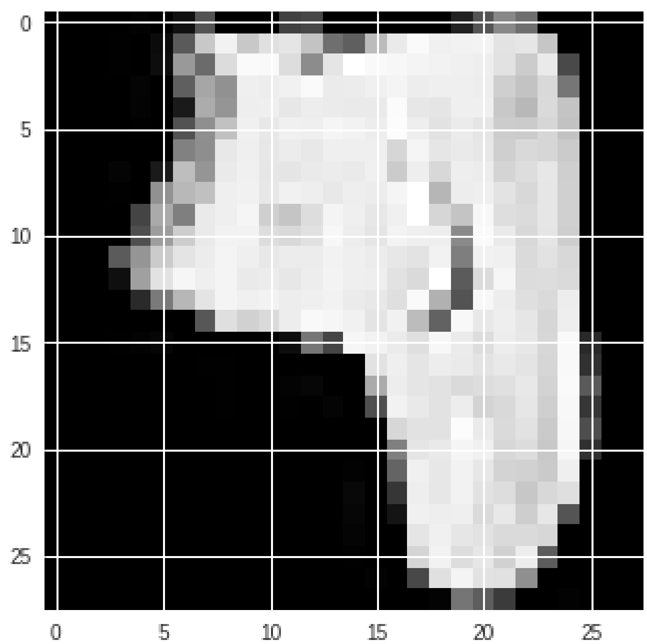
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_boot_predictions_single, test_labels, new_boot_img)
```

```
plt.subplot(1,2,2)
plot_value_array(i, new_boot_predictions_single,  test_labels)
```




```
train_images[0]

boot_train_image = train_images[0]
boot_train_image_2 = np.rot90(boot_image)
plt.imshow(boot_train_image_2.reshape(28,28), cmap='Greys_r')
```

```
<matplotlib.image.AxesImage at 0x7f0bb9467588>
```



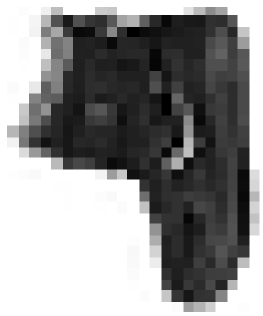

```
# Add the image to a batch where it's the only member.
new_train_boot_img = (np.expand_dims(boot_train_image_2,0))

print(new_train_boot_img.shape)
```

```
(1, 28, 28)
```

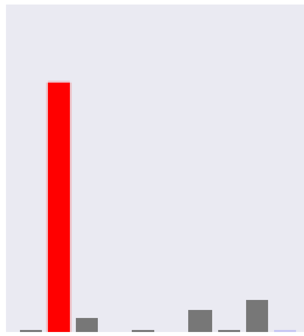
```
new_train_boot_predictions_single = model.predict(new_train_boot_img)

print(new_train_boot_predictions_single)
```

```
[[9.6188355e-03 7.5917405e-01 4.5817479e-02 1.3074232e-03 9.3391798e-03
  1.9230654e-05 7.1154073e-02 5.7603340e-03 9.7748585e-02 6.0950922e-05]]
```

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, new_train_boot_predictions_single, test_labels, new_train_boot_img)
plt.subplot(1,2,2)
plot_value_array(i, new_train_boot_predictions_single,  test_labels)
```



Trouser 76% (Ankle boot)



After rotating the images in both test data and train data the images are not predicted in right manner. The prediction is completely wrong.Therefore the accuracy is also zero.

Problem 3.

c) Download the pre-trained Inception-v3 model and run it on some images of your own choice. https://www.tensorflow.org/tutorials/images/image_recognition (https://www.tensorflow.org/tutorials/images/image_recognition) ¶

In [3]:
```
!cd /Users/aparnaaidith/Desktop/Business_DataScience_HW5/models-master/tuto
```

With an image of a flower

In [4]:
```
non /Users/aparnaaidith/Desktop/Business_DataScience_HW5/models-master/tutor
```

```
WARNING:tensorflow:From /Users/aparnaaidith/Desktop/Business_DataScience_
HW5/models-master/tutorials/image/imagenet/classify_image.py:141: FastGFi
le.__init__ (from tensorflow.python.platform.gfile) is deprecated and wil
l be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
2018-11-29 19:15:18.681387: W tensorflow/core/framework/op_def_util.cc:35
5] Op BatchNormWithGlobalNormalization is deprecated. It will cease to wo
rk in GraphDef version 9. Use tf.nn.batch_normalization().
2018-11-29 19:15:19.084735: I tensorflow/core/platform/cpu_feature_guard.
cc:141] Your CPU supports instructions that this TensorFlow binary was no
t compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
2018-11-29 19:15:19.086778: I tensorflow/core/common_runtime/process_uti
l.cc:69] Creating new thread pool with default inter op setting: 4. Tune
using inter_op_parallelism_threads for best performance.
daisy (score = 0.97947)
bee (score = 0.00099)
speedboat (score = 0.00067)
fly (score = 0.00027)
mitten (score = 0.00023)
```

With an image of a girl

In [5]:
```
!python /Users/aparnaaidith/Desktop/Business_DataScience_HW5/models-master/
```

```
WARNING:tensorflow:From /Users/aparnaaidith/Desktop/Business_DataScience_
HW5/models-master/tutorials/image/imagenet/classify_image.py:141: FastGFi
le.__init__ (from tensorflow.python.platform.gfile) is deprecated and wil
l be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
2018-11-29 19:26:28.871753: W tensorflow/core/framework/op_def_util.cc:35
5] Op BatchNormWithGlobalNormalization is deprecated. It will cease to wo
rk in GraphDef version 9. Use tf.nn.batch_normalization().
2018-11-29 19:26:29.432980: I tensorflow/core/platform/cpu_feature_guard.
cc:141] Your CPU supports instructions that this TensorFlow binary was no
t compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
2018-11-29 19:26:29.434691: I tensorflow/core/common_runtime/process_uti
l.cc:69] Creating new thread pool with default inter op setting: 4. Tune
using inter_op_parallelism_threads for best performance.
cloak (score = 0.14803)
trench coat (score = 0.14018)
suit, suit of clothes (score = 0.08267)
stole (score = 0.08201)
wool, woolen, woollen (score = 0.06680)
```

d) (Optional) Complete this tutorial for CIFAR-10. https://www.tensorflow.org/tutorials/images/deep_cnn (https://www.tensorflow.org/tutorials/images/deep_cnn)

It was taking more than 2 hours to run.Therefore stopped the execution*

In [6]:
```
!pip install tensorflow-datasets
```

```
Collecting tensorflow-datasets
  Downloading https://files.pythonhosted.org/packages/a4/02/6717354e5f6ee
ec5face466cc5a2b94f745716bea270c39301d44134abc7/tensorflow_datasets-0.0.1
-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/a4/02/6717
354e5f6eeec5face466cc5a2b94f745716bea270c39301d44134abc7/tensorflow_datas
ets-0.0.1-py2.py3-none-any.whl) (44kB)
    100% |████████████████████████████████| 51kB 2.3MB/s ta 0:00:011
Requirement already satisfied: requests in /Users/aparnaaidith/anaconda3/
lib/python3.6/site-packages (from tensorflow-datasets) (2.20.1)
Requirement already satisfied: six in /Users/aparnaaidith/anaconda3/lib/p
ython3.6/site-packages (from tensorflow-datasets) (1.11.0)
Collecting enum34 (from tensorflow-datasets)
  Downloading https://files.pythonhosted.org/packages/af/42/cb9355df32c69
b553e72a2e28daee25d1611d2c0d9c272aa1d34204205b2/enum34-1.1.6-py3-none-an
y.whl (https://files.pythonhosted.org/packages/af/42/cb9355df32c69b553e72
a2e28daee25d1611d2c0d9c272aa1d34204205b2/enum34-1.1.6-py3-none-any.whl)
Requirement already satisfied: tqdm in /Users/aparnaaidith/anaconda3/lib/
python3.6/site-packages (from tensorflow-datasets) (4.28.1)
Requirement already satisfied: future in /Users/aparnaaidith/anaconda3/li
b/python3.6/site-packages (from tensorflow-datasets) (0.17.1)
Requirement already satisfied: certifi>=2017.4.17 in /Users/aparnaaidith/
anaconda3/lib/python3.6/site-packages (from requests->tensorflow-dataset
s) (2018.10.15)
Requirement already satisfied: idna<2.8,>=2.5 in /Users/aparnaaidith/anac
onda3/lib/python3.6/site-packages (from requests->tensorflow-datasets)
 (2.7)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /Users/aparnaaidi
th/anaconda3/lib/python3.6/site-packages (from requests->tensorflow-datas
ets) (3.0.4)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /Users/aparnaaidi
th/anaconda3/lib/python3.6/site-packages (from requests->tensorflow-datas
ets) (1.23)
Installing collected packages: enum34, tensorflow-datasets
Successfully installed enum34-1.1.6 tensorflow-datasets-0.0.1
```

In [9]:
```
!python /Users/aparnaaidith/Desktop/Business_DataScience_HW5/models-master/
```

```
>> Downloading cifar-10-binary.tar.gz 100.0%
Successfully downloaded cifar-10-binary.tar.gz 170052171 bytes.
WARNING:tensorflow:From /Users/aparnaaidith/Desktop/Business_DataScience_
HW5/models-master/tutorials/image/cifar10/cifar10_input.py:158: string_in
put_producer (from tensorflow.python.training.input) is deprecated and wi
ll be removed in a future version.
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.dat
a.Dataset.from_tensor_slices(string_tensor).shuffle(tf.shape(input_tenso
r, out_type=tf.int64)[0]).repeat(num_epochs)`. If `shuffle=False`, omit t
he `.shuffle(...)`.
WARNING:tensorflow:From /Users/aparnaaidith/anaconda3/lib/python3.6/site-
packages/tensorflow/python/training/input.py:276: input_producer (from te
nsorflow.python.training.input) is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.dat
a.Dataset.from_tensor_slices(input_tensor).shuffle(tf.shape(input_tensor,
out_type=tf.int64)[0]).repeat(num_epochs)`. If `shuffle=False`, omit the
`.shuffle(...)`.
```

In [ ]:
```
tensorboard --logdir /tmp/cifar10_eval
```

In [ ]:
```
python cifar10_multi_gpu_train.py --num_gpus=2
```

Kickstarter Analysis

What makes a successful Kickstarter project?

Over two billion dollars have been raised using the massively successful crowdfunding service, Kickstarter, but not every project has found success. Of the over 300,000 projects launched on Kickstarter, only a third have made it through the funding process with a positive outcome.

Since getting funded on Kickstarter requires meeting or exceeding the project's initial goal, many organizations spend months looking through past projects in an attempt to discover some trick to finding success. Here I organize and analyze sample projects from the May 2009 to March 2017 in order to uncover any hidden trends.

Data Structure

Recently, Kickstarter released its public data repository to allow students and researchers like us to help them solve a problem. The datasets which we are planning to use is from Google data set as well as by scraping the Kickstarter website.

What are we planning to do ?

We have determined that our challenge question will investigate factors of success for a Kickstarter project as follows:

“How do KickStarter project variables impact the probability of successful funding?”

This problem will be explored through three tiers of increasing complexity. Using this system we will be able to provide a baseline of how project success by category type differs and insight on how variables impact these categories differently. This insight will culminate in a prediction model for a hypothetical project. These tiers will be attempted in the following order:

Tier 1: Identify relationship between project category and successful funding on Kickstarter

Tier 2: Stratify by project category and identify predictors of success for each

Tier 3: Build a predictive model to determine probability of funding success given hypothetical variables (potential extension of predicting probability of cancelation after successful funding)

Analysis

- Linear Regression
- Random Forest
- Seaborn
- Neural network