In [1]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\desineni\anaconda3\lib
\site-packages (2.10.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\desineni\anacond
a3\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\desineni\anaco
nda3\lib\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: numpy>=1.20 in c:\users\desineni\anaconda3\lib
\site-packages (from tensorflow) (1.21.5)
Requirement already satisfied: keras<2.11,>=2.10.0 in c:\users\desineni\anaco
nda3\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\desinen
i\anaconda3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\desineni\anaco
nda3\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\desineni\anaconda3\l
ib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\desineni\ana
conda3\lib\site-packages (from tensorflow) (3.19.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\desineni\anaco
nda3\lib\site-packages (from tensorflow) (1.42.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in c:\users\desineni\a
naconda3\lib\site-packages (from tensorflow) (2.10.1)
Requirement already satisfied: six>=1.12.0 in c:\users\desineni\anaconda3\lib
\site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\desineni\anacond
a3\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\use
rs\desineni\anaconda3\lib\site-packages (from tensorflow) (0.27.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\desineni\anaconda
3\lib\site-packages (from tensorflow) (2.0.1)
Requirement already satisfied: packaging in c:\users\desineni\anaconda3\lib\s
ite-packages (from tensorflow) (21.3)
Requirement already satisfied: setuptools in c:\users\desineni\anaconda3\lib
\site-packages (from tensorflow) (61.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\desineni\anaconda3\lib
\site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\desineni\anaconda3
\lib\site-packages (from tensorflow) (1.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\desineni
\anaconda3\lib\site-packages (from tensorflow) (4.1.1)
Requirement already satisfied: libclang>=13.0.0 in c:\users\desineni\anaconda
3\lib\site-packages (from tensorflow) (14.0.6)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in c:\users
\desineni\anaconda3\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\desineni\anaconda
3\lib\site-packages (from tensorflow) (22.9.24)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\desineni\anacon
da3\lib\site-packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\desineni\anaconda3
\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (3.3.4)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\desineni\anaconda3
\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.0.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\d
esineni\anaconda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflo
```

```
w) (0.4.6)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\desineni\anaco
nda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.27.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\desineni\ana
conda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (1.33.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\us
ers\desineni\anaconda3\lib\site-packages (from tensorboard<2.11,>=2.10->tenso
rflow) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\desi
neni\anaconda3\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow)
(1.8.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\desineni\ana
conda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10
->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\desineni\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensor
flow) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\users\desineni\an
aconda3\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.1
0->tensorflow) (4.2.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\desineni
\anaconda3\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorbo
ard<2.11,>=2.10->tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\desineni\anac
onda3\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->t
ensorboard<2.11,>=2.10->tensorflow) (0.4.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\desineni
\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.
10->tensorflow) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\desineni\anacon
da3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->ten
sorflow) (2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\desineni\ana
conda3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->
tensorflow) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\desineni\anaconda3\li
b\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflo
w) (3.3)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\desineni\anaconda3
\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>
=0.4.1->tensorboard<2.11,>=2.10->tensorflow) (3.2.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\desineni
\anaconda3\lib\site-packages (from packaging->tensorflow) (3.0.4)
```

In [2]:
```
!pip install keras
```

```
Requirement already satisfied: keras in c:\users\desineni\anaconda3\lib\site-pa
ckages (2.10.0)
```

In [46]:
```
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

In [47]:
```
train_data[0]
```

Out[47]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
```

In [48]:
```
train_labels[0]
```

Out[48]:
```
1
```

In [49]:
```
max([max(sequence) for sequence in train_data])
```

Out[49]:
```
9999
```

In [50]:
```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = " ".join(
    [reverse_word_index.get(i - 3, "?") for i in train_data[0]])
```

In [51]:
```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for j in sequence:
            results[i, j] = 1.
    return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [52]:
```
x_train[0]
```

Out[52]:
```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [53]:
```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

In [54]:
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

Step 1 :

1. Sequential Three layered approach
2. Replaced relu with tanh
3. optimizers changed to adam and loss to mse and metrics == accuracy

In [70]:
```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense

model = keras.Sequential()
model.add(Dense(16, activation="tanh"))
model.add(Dense(16, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))


model.compile(optimizer="adam",
              loss="mean_squared_error",
              metrics=["accuracy"])
```

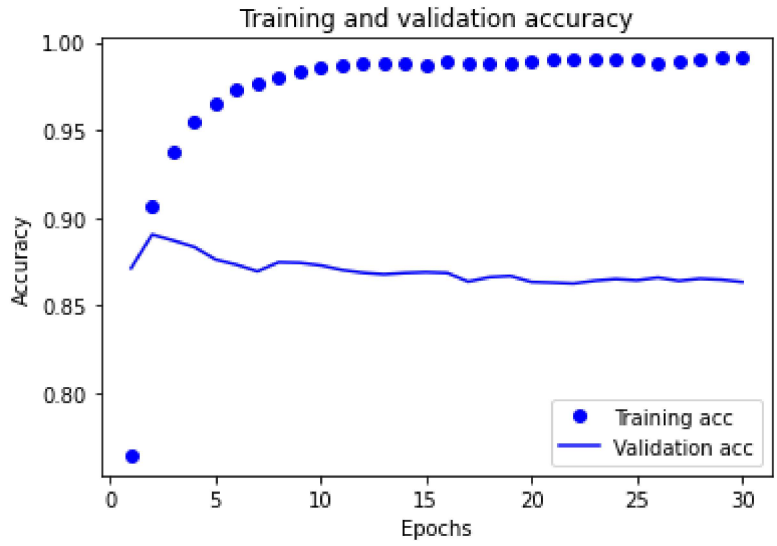
In [71]:
```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1667 - accura
cy: 0.7984 - val_loss: 0.1114 - val_accuracy: 0.8654
Epoch 2/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0778 - accura
cy: 0.9121 - val_loss: 0.0865 - val_accuracy: 0.8872
Epoch 3/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0509 - accura
cy: 0.9445 - val_loss: 0.0836 - val_accuracy: 0.8872
Epoch 4/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0360 - accura
cy: 0.9643 - val_loss: 0.0845 - val_accuracy: 0.8857
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0264 - accura
cy: 0.9769 - val_loss: 0.0870 - val_accuracy: 0.8826
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0197 - accura
cy: 0.9847 - val_loss: 0.0909 - val_accuracy: 0.8791
Epoch 7/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0154 - accura
cy: 0.9881 - val_loss: 0.0928 - val_accuracy: 0.8768
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0119 - accura
cy: 0.9914 - val_loss: 0.0957 - val_accuracy: 0.8748
Epoch 9/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0096 - accura
cy: 0.9929 - val_loss: 0.0977 - val_accuracy: 0.8747
Epoch 10/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0081 - accura
cy: 0.9941 - val_loss: 0.1000 - val_accuracy: 0.8730
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0069 - accura
cy: 0.9949 - val_loss: 0.1018 - val_accuracy: 0.8709
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0060 - accura
cy: 0.9954 - val_loss: 0.1032 - val_accuracy: 0.8690
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0055 - accura
cy: 0.9956 - val_loss: 0.1044 - val_accuracy: 0.8688
Epoch 14/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0052 - accura
cy: 0.9957 - val_loss: 0.1050 - val_accuracy: 0.8700
Epoch 15/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0049 - accura
cy: 0.9959 - val_loss: 0.1058 - val_accuracy: 0.8703
Epoch 16/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0046 - accura
cy: 0.9959 - val_loss: 0.1066 - val_accuracy: 0.8698
Epoch 17/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0044 - accura
```

```
cy: 0.9961 - val_loss: 0.1080 - val_accuracy: 0.8675
Epoch 18/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0043 - accura
cy: 0.9961 - val_loss: 0.1080 - val_accuracy: 0.8677
Epoch 19/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0042 - accura
cy: 0.9962 - val_loss: 0.1083 - val_accuracy: 0.8679
Epoch 20/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0041 - accura
cy: 0.9962 - val_loss: 0.1089 - val_accuracy: 0.8670
```

In [57]:

```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



In [58]:
```
# Step 2

### implement dropouts and Regularizers
### check performance by changing the dense layers to 64 hidden units

from tensorflow.keras.layers import Dropout
from tensorflow.keras import regularizers


model = keras.Sequential()
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dense(1, activation="sigmoid",activity_regularizer=regularizers.L2(0.01


model.compile(optimizer="adam",
              loss="mean_squared_error",
              metrics=["accuracy"])

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=30,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/30
30/30 [==============================] - 1s 27ms/step - loss: 0.1577 - accurac
y: 0.7748 - val_loss: 0.0918 - val_accuracy: 0.8819
Epoch 2/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0722 - accurac
y: 0.9091 - val_loss: 0.0908 - val_accuracy: 0.8830
Epoch 3/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0548 - accurac
y: 0.9350 - val_loss: 0.0928 - val_accuracy: 0.8817
Epoch 4/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0436 - accurac
y: 0.9515 - val_loss: 0.0960 - val_accuracy: 0.8826
Epoch 5/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0363 - accurac
y: 0.9612 - val_loss: 0.1040 - val_accuracy: 0.8761
Epoch 6/30
30/30 [==============================] - 1s 22ms/step - loss: 0.0312 - accurac
y: 0.9681 - val_loss: 0.1076 - val_accuracy: 0.8735
Epoch 7/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0292 - accurac
y: 0.9705 - val_loss: 0.1097 - val_accuracy: 0.8742
Epoch 8/30
30/30 [==============================] - 1s 21ms/step - loss: 0.0266 - accurac
y: 0.9746 - val_loss: 0.1133 - val_accuracy: 0.8716
Epoch 9/30
30/30 [==============================] - 1s 22ms/step - loss: 0.0236 - accurac
y: 0.9785 - val_loss: 0.1175 - val_accuracy: 0.8707
Epoch 10/30
```

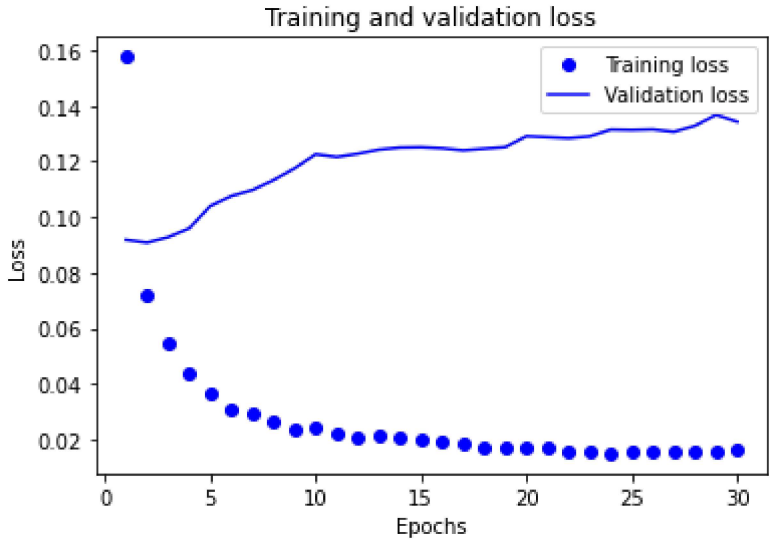
```
30/30 [==============================] - 1s 21ms/step - loss: 0.0240 - accurac
y: 0.9770 - val_loss: 0.1226 - val_accuracy: 0.8636
Epoch 11/30
30/30 [==============================] - 1s 21ms/step - loss: 0.0219 - accurac
y: 0.9808 - val_loss: 0.1216 - val_accuracy: 0.8671
Epoch 12/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0207 - accurac
y: 0.9811 - val_loss: 0.1228 - val_accuracy: 0.8658
Epoch 13/30
30/30 [==============================] - 1s 21ms/step - loss: 0.0213 - accurac
y: 0.9811 - val_loss: 0.1243 - val_accuracy: 0.8650
Epoch 14/30
30/30 [==============================] - 1s 20ms/step - loss: 0.0208 - accurac
y: 0.9816 - val_loss: 0.1250 - val_accuracy: 0.8662
Epoch 15/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0201 - accurac
y: 0.9822 - val_loss: 0.1251 - val_accuracy: 0.8662
Epoch 16/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0197 - accurac
y: 0.9831 - val_loss: 0.1247 - val_accuracy: 0.8675
Epoch 17/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0184 - accurac
y: 0.9848 - val_loss: 0.1240 - val_accuracy: 0.8678
Epoch 18/30
30/30 [==============================] - 1s 20ms/step - loss: 0.0174 - accurac
y: 0.9863 - val_loss: 0.1246 - val_accuracy: 0.8678
Epoch 19/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0173 - accurac
y: 0.9861 - val_loss: 0.1252 - val_accuracy: 0.8679
Epoch 20/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0174 - accurac
y: 0.9859 - val_loss: 0.1291 - val_accuracy: 0.8637
Epoch 21/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0175 - accurac
y: 0.9861 - val_loss: 0.1287 - val_accuracy: 0.8639
Epoch 22/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0160 - accurac
y: 0.9880 - val_loss: 0.1284 - val_accuracy: 0.8654
Epoch 23/30
30/30 [==============================] - 1s 20ms/step - loss: 0.0157 - accurac
y: 0.9882 - val_loss: 0.1290 - val_accuracy: 0.8651
Epoch 24/30
30/30 [==============================] - 1s 19ms/step - loss: 0.0151 - accurac
y: 0.9892 - val_loss: 0.1315 - val_accuracy: 0.8634
Epoch 25/30
30/30 [==============================] - 1s 20ms/step - loss: 0.0157 - accurac
y: 0.9882 - val_loss: 0.1313 - val_accuracy: 0.8633
Epoch 26/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0156 - accurac
y: 0.9882 - val_loss: 0.1315 - val_accuracy: 0.8621
Epoch 27/30
30/30 [==============================] - 1s 24ms/step - loss: 0.0156 - accurac
y: 0.9884 - val_loss: 0.1307 - val_accuracy: 0.8639
Epoch 28/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0155 - accurac
y: 0.9883 - val_loss: 0.1328 - val_accuracy: 0.8621
Epoch 29/30
```

```
30/30 [==============================] - 1s 28ms/step - loss: 0.0160 - accurac
y: 0.9881 - val_loss: 0.1368 - val_accuracy: 0.8575
Epoch 30/30
30/30 [==============================] - 1s 26ms/step - loss: 0.0162 - accurac
y: 0.9872 - val_loss: 0.1343 - val_accuracy: 0.8613
```

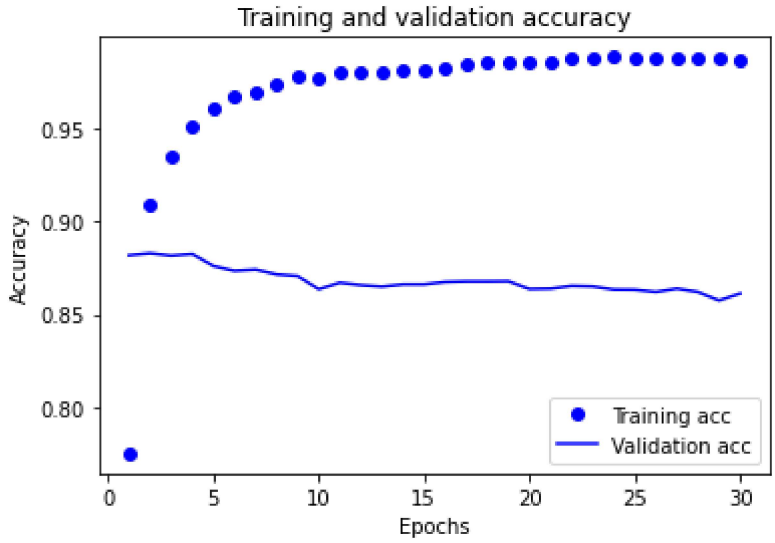
In [59]:
```
history_dict = history.history
history_dict.keys()
```

Out[59]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [60]:
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



In [61]:
```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



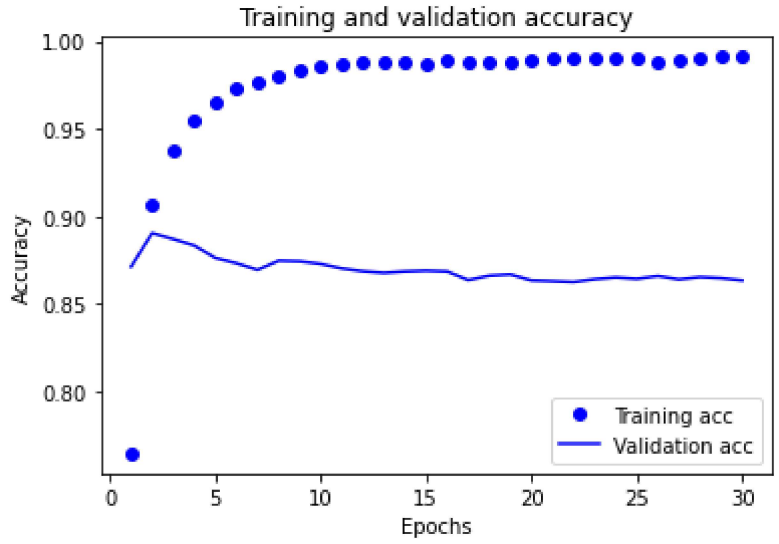
In [63]:
```
from tensorflow.keras.layers import Dropout
from tensorflow.keras import regularizers


model = keras.Sequential()
model.add(Dense(64, activation="tanh"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh",activity_regularizer=regularizers.L2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(64, activation="tanh"))
model.add(Dense(64, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))


model.compile(optimizer="adam",
              loss="mean_squared_error",
              metrics=["accuracy"])

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=30,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/30
30/30 [==============================] - 2s 55ms/step - loss: 0.1747 - accurac
y: 0.7810 - val_loss: 0.1038 - val_accuracy: 0.8846
Epoch 2/30
30/30 [==============================] - 1s 38ms/step - loss: 0.0845 - accurac
y: 0.9148 - val_loss: 0.0954 - val_accuracy: 0.8852
Epoch 3/30
30/30 [==============================] - 1s 30ms/step - loss: 0.0608 - accurac
y: 0.9402 - val_loss: 0.0982 - val_accuracy: 0.8802
Epoch 4/30
30/30 [==============================] - 1s 29ms/step - loss: 0.0486 - accurac
y: 0.9518 - val_loss: 0.1003 - val_accuracy: 0.8784
Epoch 5/30
30/30 [==============================] - 1s 27ms/step - loss: 0.0392 - accurac
y: 0.9609 - val_loss: 0.1071 - val_accuracy: 0.8704
Epoch 6/30
30/30 [==============================] - 1s 29ms/step - loss: 0.0324 - accurac
y: 0.9669 - val_loss: 0.1082 - val_accuracy: 0.8734
Epoch 7/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0298 - accurac
y: 0.9705 - val_loss: 0.1152 - val_accuracy: 0.8698
Epoch 8/30
30/30 [==============================] - 1s 23ms/step - loss: 0.0289 - accurac
y: 0.9700 - val_loss: 0.1170 - val_accuracy: 0.8676
Epoch 9/30
30/30 [==============================] - 1s 23ms/step - loss: 0.0264 - accurac
y: 0.9731 - val_loss: 0.1189 - val_accuracy: 0.8662
Epoch 10/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0244 - accurac
y: 0.9767 - val_loss: 0.1202 - val_accuracy: 0.8651
```

```
Epoch 11/30
30/30 [==============================] - 1s 27ms/step - loss: 0.0227 - accurac
y: 0.9779 - val_loss: 0.1206 - val_accuracy: 0.8658
Epoch 12/30
30/30 [==============================] - 1s 27ms/step - loss: 0.0230 - accurac
y: 0.9770 - val_loss: 0.1227 - val_accuracy: 0.8639
Epoch 13/30
30/30 [==============================] - 1s 24ms/step - loss: 0.0203 - accurac
y: 0.9796 - val_loss: 0.1233 - val_accuracy: 0.8657
Epoch 14/30
30/30 [==============================] - 1s 23ms/step - loss: 0.0203 - accurac
y: 0.9803 - val_loss: 0.1236 - val_accuracy: 0.8647
Epoch 15/30
30/30 [==============================] - 1s 24ms/step - loss: 0.0185 - accurac
y: 0.9815 - val_loss: 0.1249 - val_accuracy: 0.8624
Epoch 16/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0175 - accurac
y: 0.9833 - val_loss: 0.1262 - val_accuracy: 0.8629
Epoch 17/30
30/30 [==============================] - 1s 23ms/step - loss: 0.0200 - accurac
y: 0.9797 - val_loss: 0.1269 - val_accuracy: 0.8621
Epoch 18/30
30/30 [==============================] - 1s 30ms/step - loss: 0.0185 - accurac
y: 0.9816 - val_loss: 0.1289 - val_accuracy: 0.8619
Epoch 19/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0171 - accurac
y: 0.9831 - val_loss: 0.1315 - val_accuracy: 0.8596
Epoch 20/30
30/30 [==============================] - 1s 29ms/step - loss: 0.0179 - accurac
y: 0.9821 - val_loss: 0.1267 - val_accuracy: 0.8653
Epoch 21/30
30/30 [==============================] - 1s 29ms/step - loss: 0.0180 - accurac
y: 0.9814 - val_loss: 0.1276 - val_accuracy: 0.8630
Epoch 22/30
30/30 [==============================] - 1s 27ms/step - loss: 0.0172 - accurac
y: 0.9829 - val_loss: 0.1267 - val_accuracy: 0.8626
Epoch 23/30
30/30 [==============================] - 1s 28ms/step - loss: 0.0178 - accurac
y: 0.9824 - val_loss: 0.1299 - val_accuracy: 0.8597
Epoch 24/30
30/30 [==============================] - 1s 26ms/step - loss: 0.0156 - accurac
y: 0.9848 - val_loss: 0.1355 - val_accuracy: 0.8565
Epoch 25/30
30/30 [==============================] - 1s 26ms/step - loss: 0.0158 - accurac
y: 0.9844 - val_loss: 0.1280 - val_accuracy: 0.8626
Epoch 26/30
30/30 [==============================] - 1s 27ms/step - loss: 0.0156 - accurac
y: 0.9853 - val_loss: 0.1274 - val_accuracy: 0.8643
Epoch 27/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0144 - accurac
y: 0.9859 - val_loss: 0.1335 - val_accuracy: 0.8560
Epoch 28/30
30/30 [==============================] - 1s 26ms/step - loss: 0.0160 - accurac
y: 0.9848 - val_loss: 0.1302 - val_accuracy: 0.8616
Epoch 29/30
30/30 [==============================] - 1s 25ms/step - loss: 0.0155 - accurac
y: 0.9850 - val_loss: 0.1302 - val_accuracy: 0.8624
```

```
Epoch 30/30
30/30 [==============================] - 1s 24ms/step - loss: 0.0161 - accurac
y: 0.9835 - val_loss: 0.1317 - val_accuracy: 0.8612
```

In [44]:
```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



Summary

Significance:
My approach towards the problem, initially learned the importance of keras Sequential model which is stack of layers for building neural network.
It contains the important imports like, layers, Dense, Dropouts and Regularizers required to design a neural network
These are required imports

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras import regularizers
```

- I implemented 2, 3 layered and 6 layered neural network with 16 ,64, 64 hidden neurons to check the performance.
- One important thing that I observed is no matter how many layers we stacked up, it will train and gives the almost same performance once it reaches the threshold.

```
model = keras.Sequential()
```

it follows the structure like input layer hidden layers output layer

The above line initializes the Sequential model where we can build up stack of layers.

model.add(Dense(64, activation="tanh"))

- The above line signifies that we are adding a new hidden layer with 64 dense units using tanh activation function.
- When I say 64 hidden units, we can assume that they are 64 neuron are being created in the layer to learn the data which is in form of vectors
- Activation function is also called transfer function, if the output range of function is limited, just sigmoid squashed the value to 1 which is above to it. They are non linear functions.

model.add(Dropout(0.5))

- The importance of Dropout is useful when we have a scenario of overfitting. The above line says, hey please randomly dropout some of my neurons since it causes overfitting. When I say 0.5, that implies 50 percent of my neurons are dropped out.
- I have tried using L1 and L2 regularizes but that does not cause much effect rather it diminishes performance. I think the model Is saturated and the best validation accuracy we can get a range of 86 or 87 percentage.
- Replaced binary_crossentrophy with mean square error to check the performance metrics on the loss.
- The result is that validation loss has performed good. Initially when using binary_crossentrophy the validation loss is 0.5 but when using mse it reduces to 0.1.
- Relu being the top among sigmoid and tanh function because of vanishing gradient problem. In this context tanh performs same as tanh.

Combinations	Training accuracy	Validation accuracy
2 dense layers 16 hidden units Tanh activation function Optimizers = adam	99.63	87.01
3 dense layers 64 hidden units Tanh activation function Dropouts (0.5) Regularizers Optimizers = adam	98	86.73
6 dense layers 64 hidden units Tanh activation function Dropouts (0.5) Optimizers = adam regularizers	98.75	86.12

In [ ]: