Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 2
Pembahasan : Implementasi ANN KOTLIN
Pokok Pemb : Implementasi ANN dengan Tensorflow Lite dan Kotlin

Import Library dan membuat Array

```
import tensorflow as tf
import numpy as np
from tensorflow import keras
import matplotlib.pyplot as plt
```

```
jarak = np.array([2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0], dtype=float)
bbm = np.array([6.0, 10.0, 14.0, 18.0, 22.0, 26.0, 30.0], dtype=float)
```

```
plt.plot(jarak, bbm, 'o-')
plt.xlabel('jarak')
plt.ylabel('bbm')
plt.title('Plot jarak vs bbm')
plt.grid(True)
plt.show()
```

Membuat model dengan ANN

```
model = tf.keras.Sequential([
    keras.layers.Dense(units=1, input_shape=[1])])
```

```
model.compile(
    optimizer='sgd',
    loss='mean_squared_error')
```

Menjalankan model dan simulasi data baru

```
model.fit(jarak, bbm, epochs=150)
```

```
print(model.predict(np.array([13.0])))
```

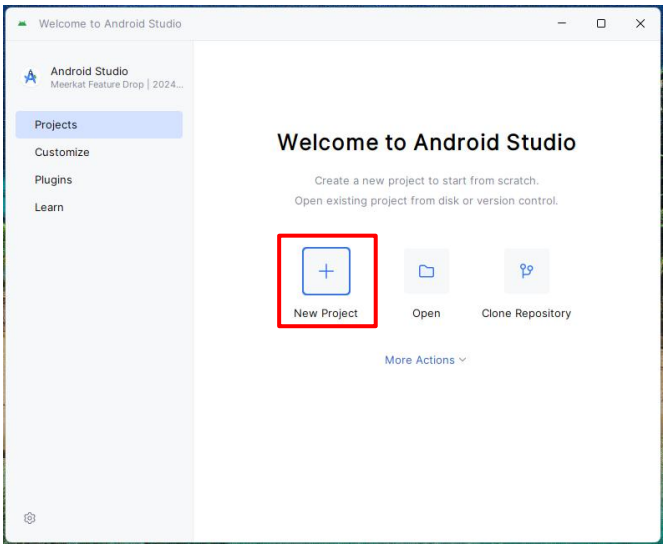
```
# membuat prediksi dengan plot
predicted_bbm = model.predict(jarak)

# Plot Dataset
plt.figure(figsize=(10, 6))
plt.plot(jarak, bbm, 'o-', label='Dataset')

# Plot hasil prediksi
plt.plot(jarak, predicted_bbm, 'x--', label='Prediksi')

# Plot nilai baru
new_jarak = np.array([13.0])
predicted_new_bbm = model.predict(new_jarak)
plt.plot(new_jarak, predicted_new_bbm, 'ro', markersize=8, label='Hasil Prediksi')


plt.xlabel('jarak')
plt.ylabel('bbm')
plt.title('Plot jarak vs bbm dengan Prediksi')
plt.grid(True)
plt.legend()
plt.show()
```

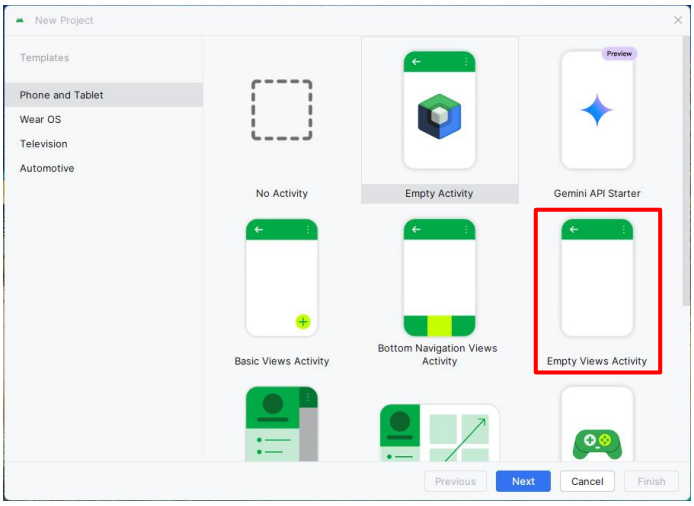
Menyimpan Model

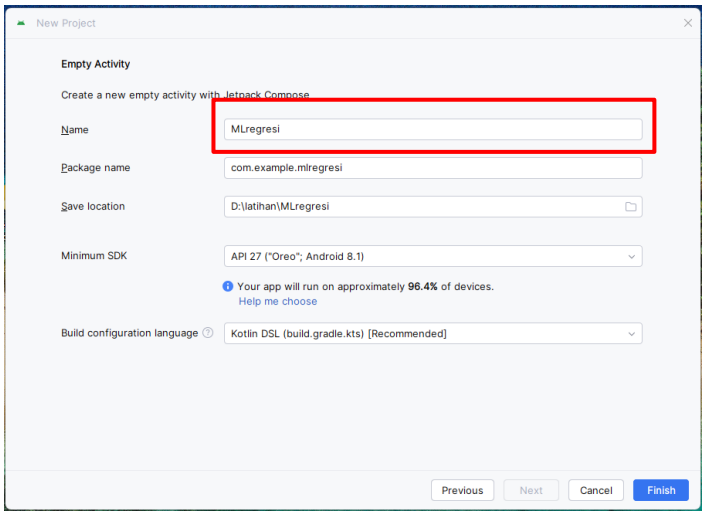
```
keras_file = "linear.h5"
tf.keras.models.save_model(model, keras_file)
```

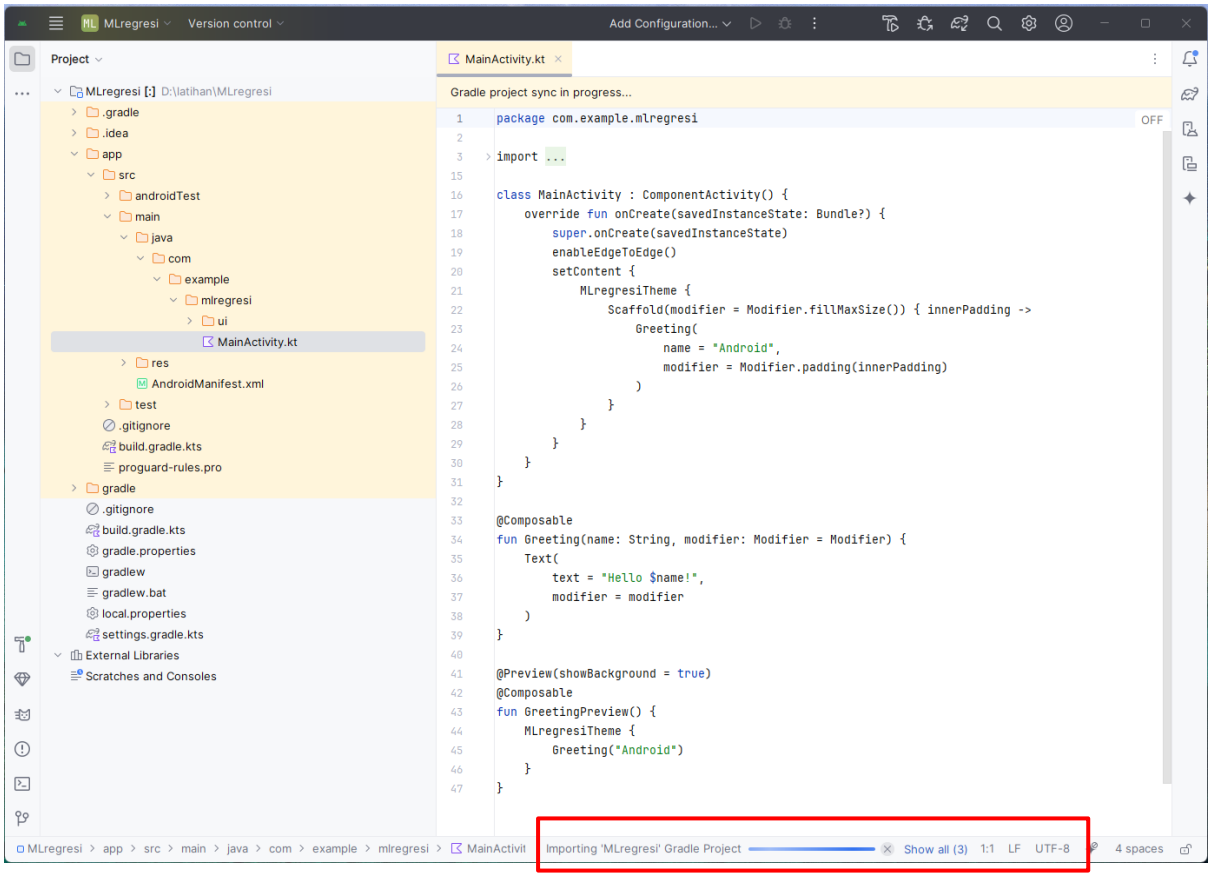
```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
open("linear.tflite", "wb").write(tflite_model)
```

Membuat project baru di Android Studio



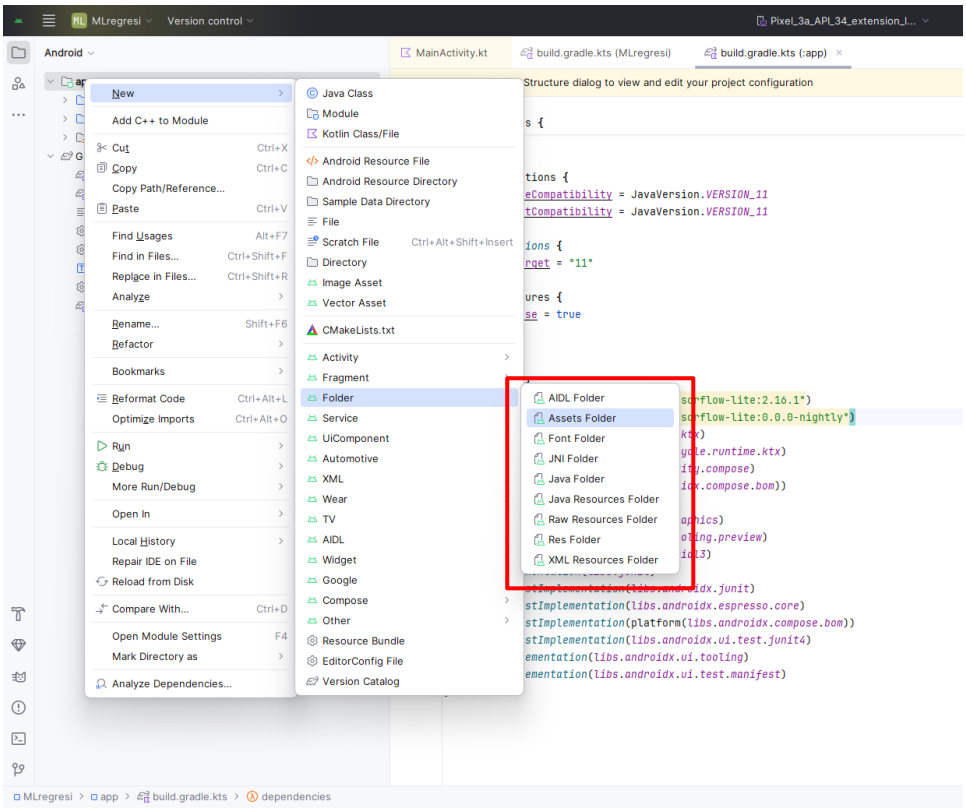


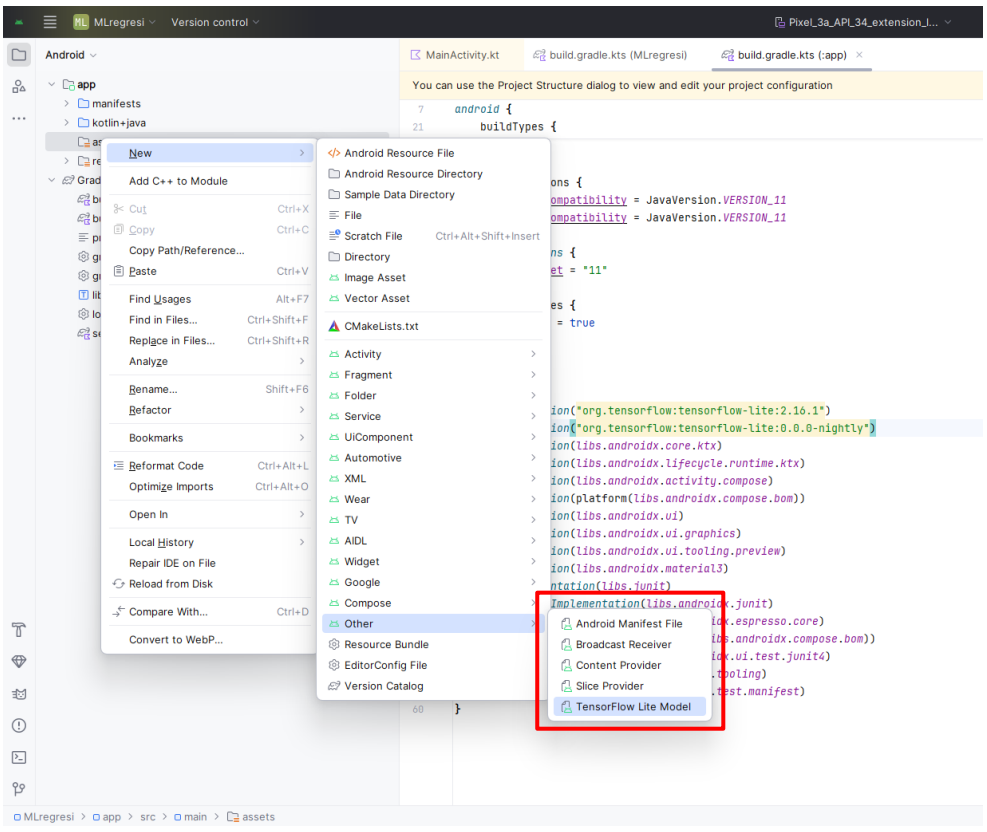


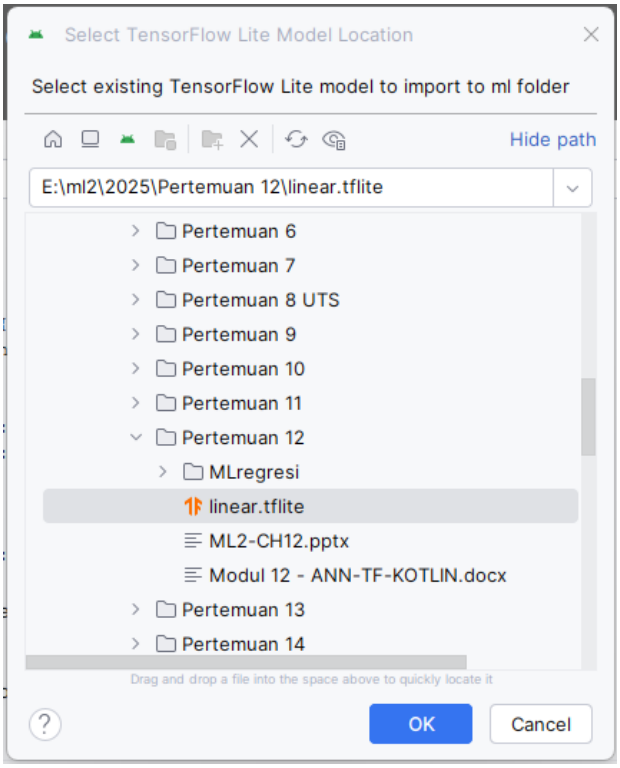


Menambahkan tensorflow package pada build.gradle

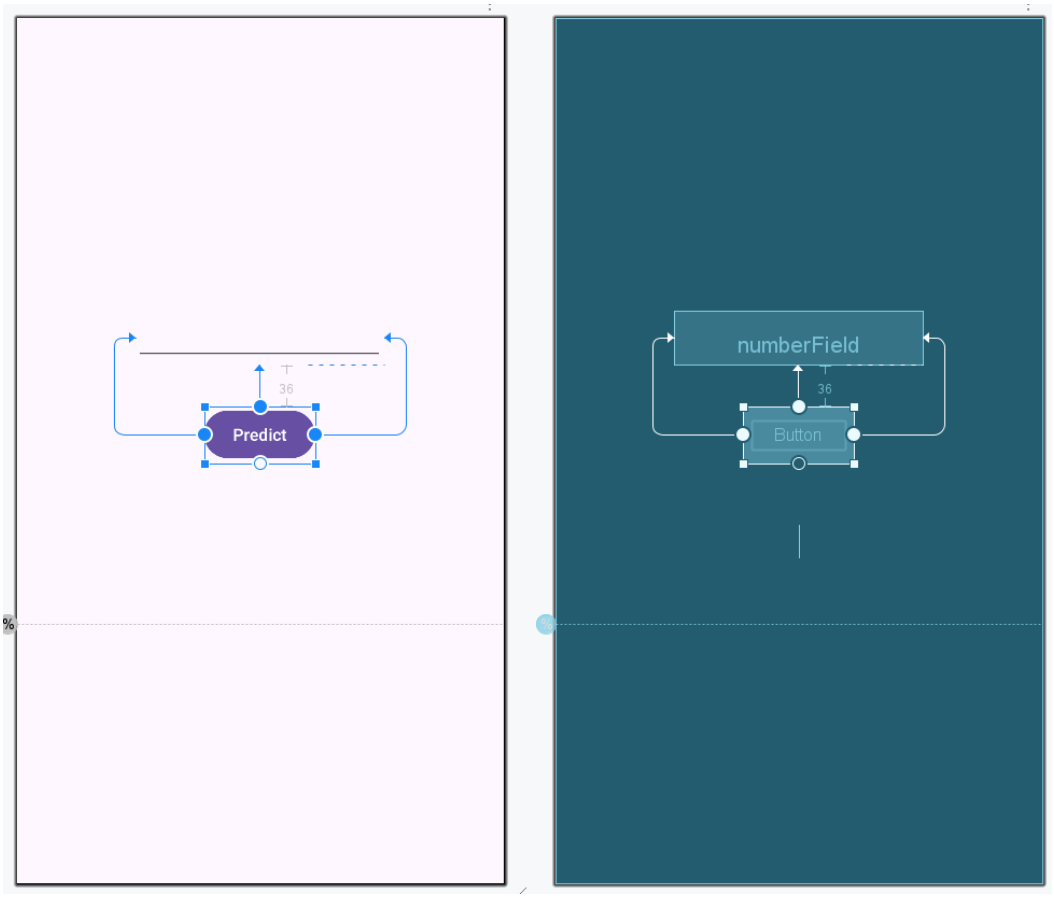
Menambahkan folder assets untuk menyimpan model







Membuat Layout Aplikasi



activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.7" />

    <EditText
        android:id="@+id/numberField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="220dp"
        android:ems="10"
        android:inputType="number"
        app:layout_constraintBottom_toTopOf="@id/guideline"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="MissingConstraints" />

    <Button
        android:id="@+id/btnPredict"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="36dp"
        android:text="Predict"
        app:layout_constraintEnd_toEndOf="@+id/numberField"
        app:layout_constraintStart_toStartOf="@+id/numberField"
        app:layout_constraintTop_toBottomOf="@+id/numberField"
        tools:ignore="MissingConstraints" />

    <TextView
        android:id="@+id/txtResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="52dp"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="@+id/btnPredict"
        app:layout_constraintStart_toStartOf="@+id/btnPredict"
        app:layout_constraintTop_toBottomOf="@+id/btnPredict" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Library yang dibutuhkan dalam aplikasi

MainActivity.kt

```
package com.example.mlregresi

import android.content.res.AssetManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel
```

Menentukan parameter nilai input dan output

MainActivity.kt

```
private lateinit var interpreter: Interpreter
private val mModelPath = "linear.tflite"

private lateinit var resultText: TextView
private lateinit var editText: EditText
private lateinit var checkButton: Button

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    resultText = findViewById(R.id.txtResult)
    editText = findViewById(R.id.numberField)
    checkButton = findViewById(R.id.btnPredict)

    checkButton.setOnClickListener{
        var result = doInference(editText.text.toString())
        runOnUiThread{
            resultText.text = result.toString()
        }
    }

    initInterpreter()
}
```

Membuat Interpreter

MainActivity.kt

```
private fun initInterpreter(){
    val options = Interpreter.Options()
    options.setNumThreads(4)
    options.setUseNNAPI(true)
    interpreter = Interpreter(loadModelFile(assets, mModelPath), options)
}
```

Menjalankan Interpreter dengan data input dan output

MainActivity.kt

```
private fun doInference(inputString: String): Float{
    val inputVal = FloatArray( size: 1)
    inputVal[0] = inputString.toFloat()
    val ouput = Array( size: 1) {FloatArray( size: 1)}
    interpreter.run(inputVal, ouput)
    return ouput[0][0]
}
```

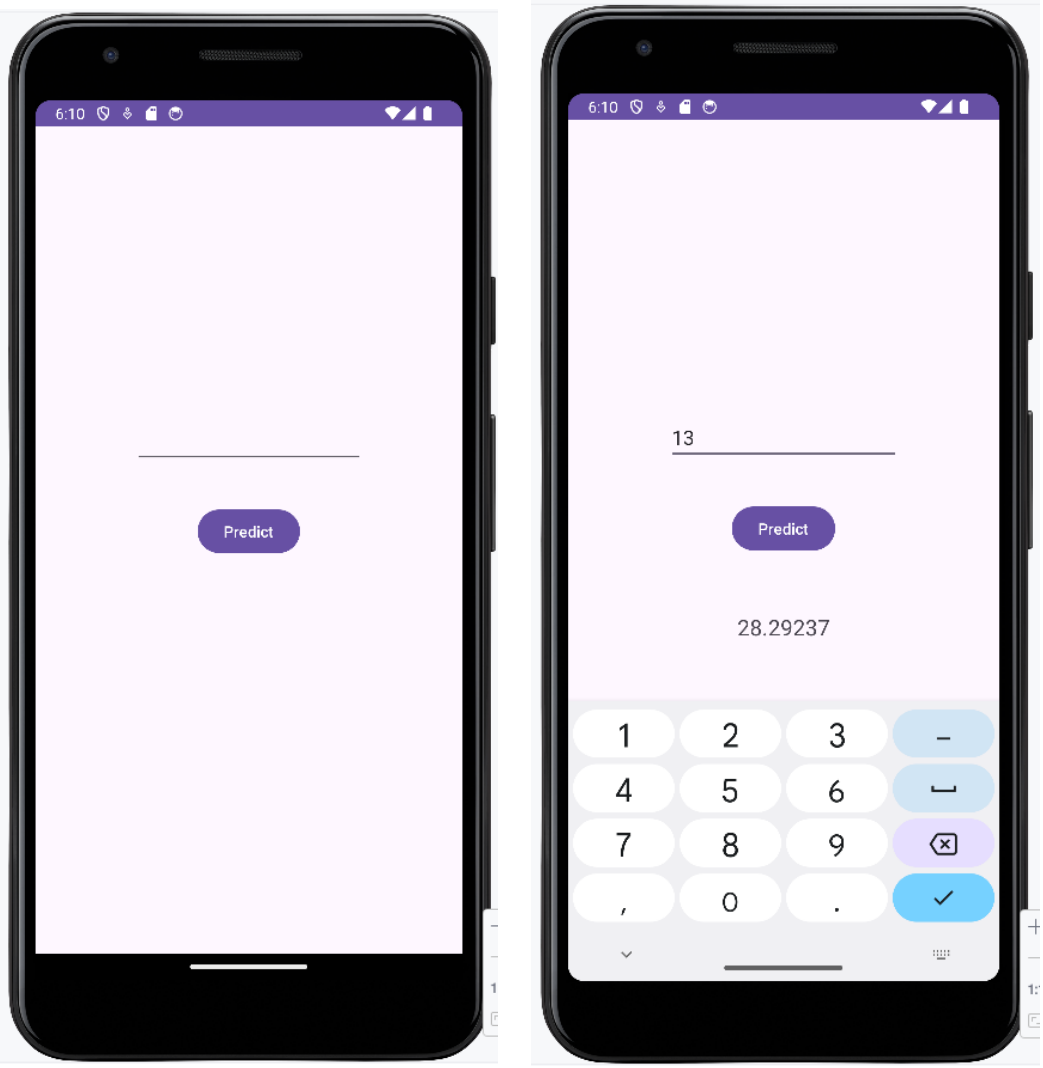
Membuat fungsi untuk menjalankan model

MainActivity.kt

```
private fun loadModelFile(assetManager: AssetManager, modelPath: String): MappedByteBuffer{
    val fileDescriptor = assetManager.openFd(modelPath)
    val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
    val fileChannel = inputStream.channel
    val startOffset = fileDescriptor.startOffset
    val declaredLength = fileDescriptor.declaredLength
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength)
}
```

MainActivity.kt

```
package com.example.mlregresi

import android.content.res.AssetManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel

class MainActivity : AppCompatActivity(){

    private lateinit var interpreter: Interpreter
    private val mModelPath = "linear.tflite"

    private lateinit var resultText: TextView
    private lateinit var editText: EditText
    private lateinit var checkButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        resultText = findViewById(R.id.txtResult)
        editText = findViewById(R.id.numberField)
        checkButton = findViewById(R.id.btnPredict)

        checkButton.setOnClickListener{
            var result = doInference(editText.text.toString())
            runOnUiThread{
                resultText.text = result.toString()
            }
        }

        initInterpreter()
    }

    private fun initInterpreter(){
        val options = Interpreter.Options()
        options.setNumThreads(4)
        options.setUseNNAPI(true)
        interpreter = Interpreter(loadModelFile(assets, mModelPath), options)
    }

    private fun doInference(inputString: String): Float{
        val inputVal = FloatArray( size: 1)
        inputVal[0] = inputString.toFloat()
        val ouput = Array( size: 1) {FloatArray( size: 1)}
        interpreter.run(inputVal, ouput)
        return ouput[0][0]
    }

    private fun loadModelFile(assetManager: AssetManager, modelPath: String): MappedByteBuffer{
        val fileDescriptor = assetManager.openFd(modelPath)
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declaredLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength)
    }

}
```


6:10
Predict
6:10
13
Predict
28.29237
1
2
3
4
5
6
7
8
9
,
0
.