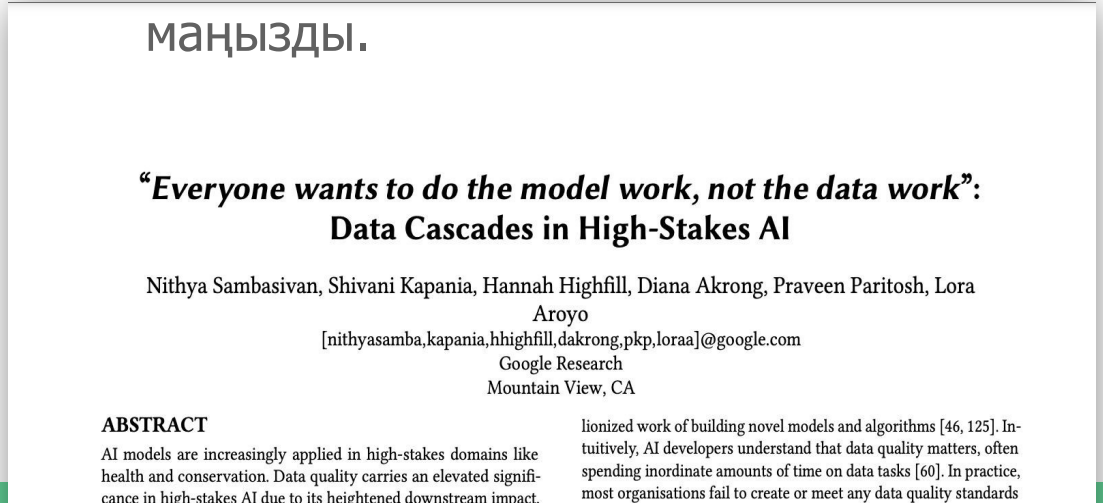
ML жобаларын қалай ұйымдастыруға болады

MLSys: Пайдалану жағдайлары

- Модельдер ML платформаларының кішкене бөлігі болып табылады және көбінесе ең аз проблемалы (кейбір ескертулермен);
- барлығы модель жұмысын орындағысы келгенімен, деректер жұмысы жиі бірдей болады (немесе одан да көп) тәжірибеде маңызды.



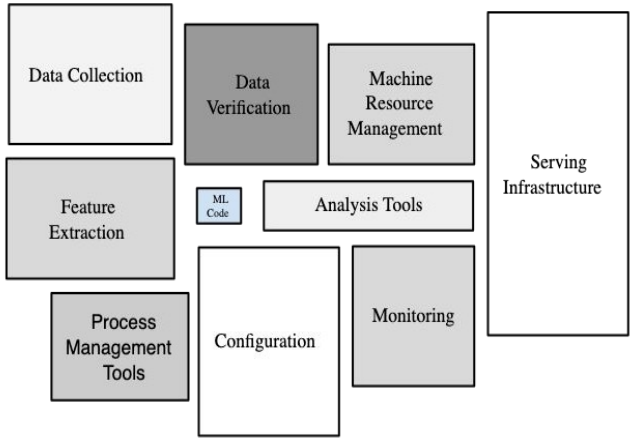
“*Everyone wants to do the model work, not the data work*”: Data Cascades in High-Stakes AI

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, Lora Aroyo
[nithyasamba,kapania,hhighfill,dakrong,pkp,loraa]@google.com
Google Research
Mountain View, CA

ABSTRACT

AI models are increasingly applied in high-stakes domains like health and conservation. Data quality carries an elevated significance in high-stakes AI due to its heightened downstream impact.

lionized work of building novel models and algorithms [46, 125]. Intuitively, AI developers understand that data quality matters, often spending inordinate amounts of time on data tasks [60]. In practice, most organisations fail to create or meet any data quality standards



ML жобаларының үш негізгі кезеңі

Деректер	*Оқыту*	*Қорытынды*
• Жиналу • Тазалау • Тестілеу • Кодтау • ….	• Модельдеу • Гиперпарамдық баптау • Тестілеу • …	• Қызмет көрсету • Кэштеу • Бақылау • ….

ML жобаларының үш негізгі кезеңі

Деректер

Оқыту

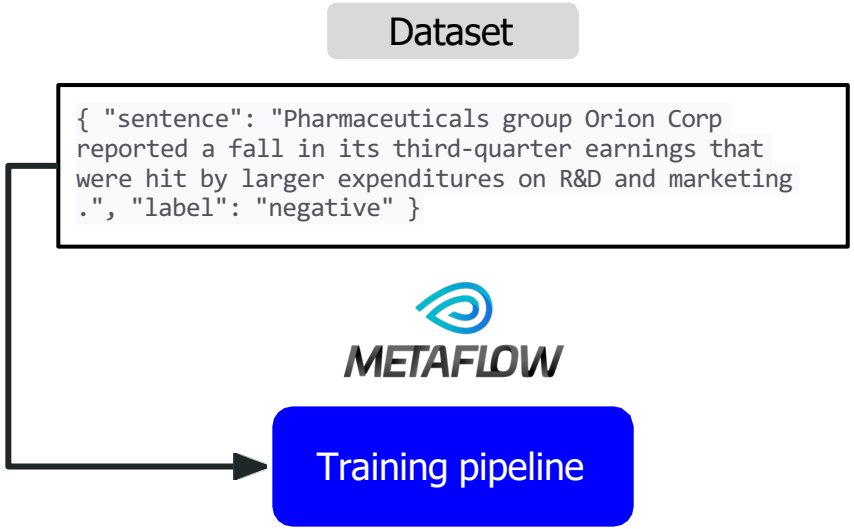
Қорытынды

Ноутбукта *

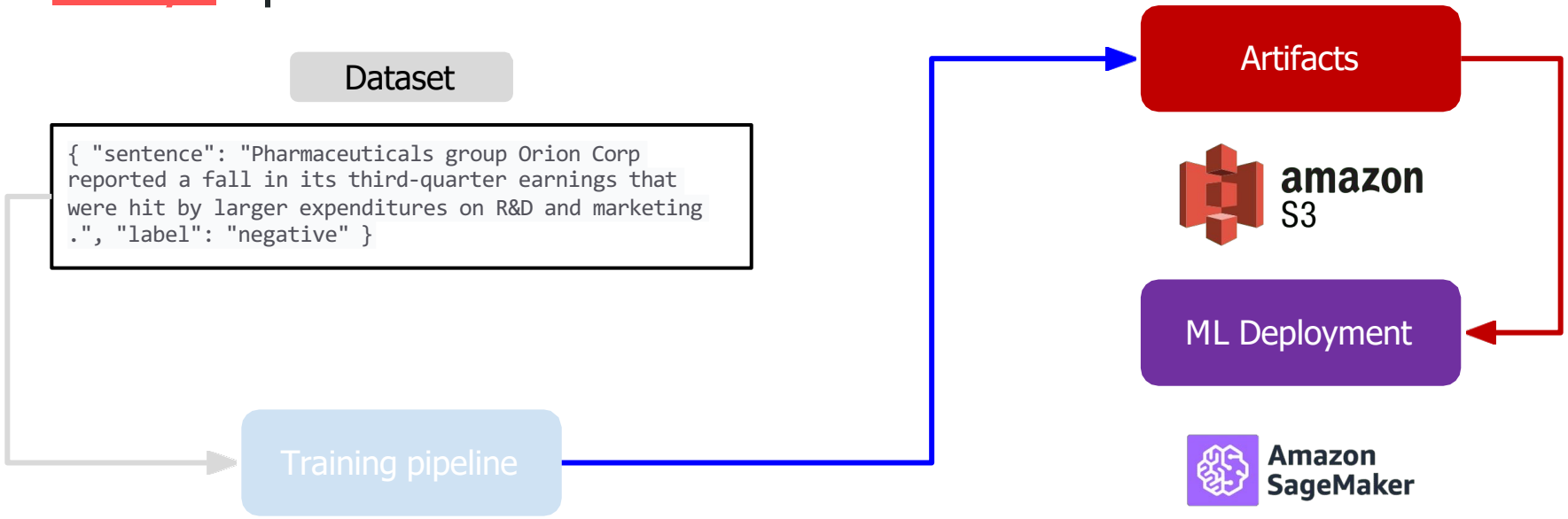
бұлтта

*Conditions apply. In particular, Metaflow sandboxes are *also cloud!*

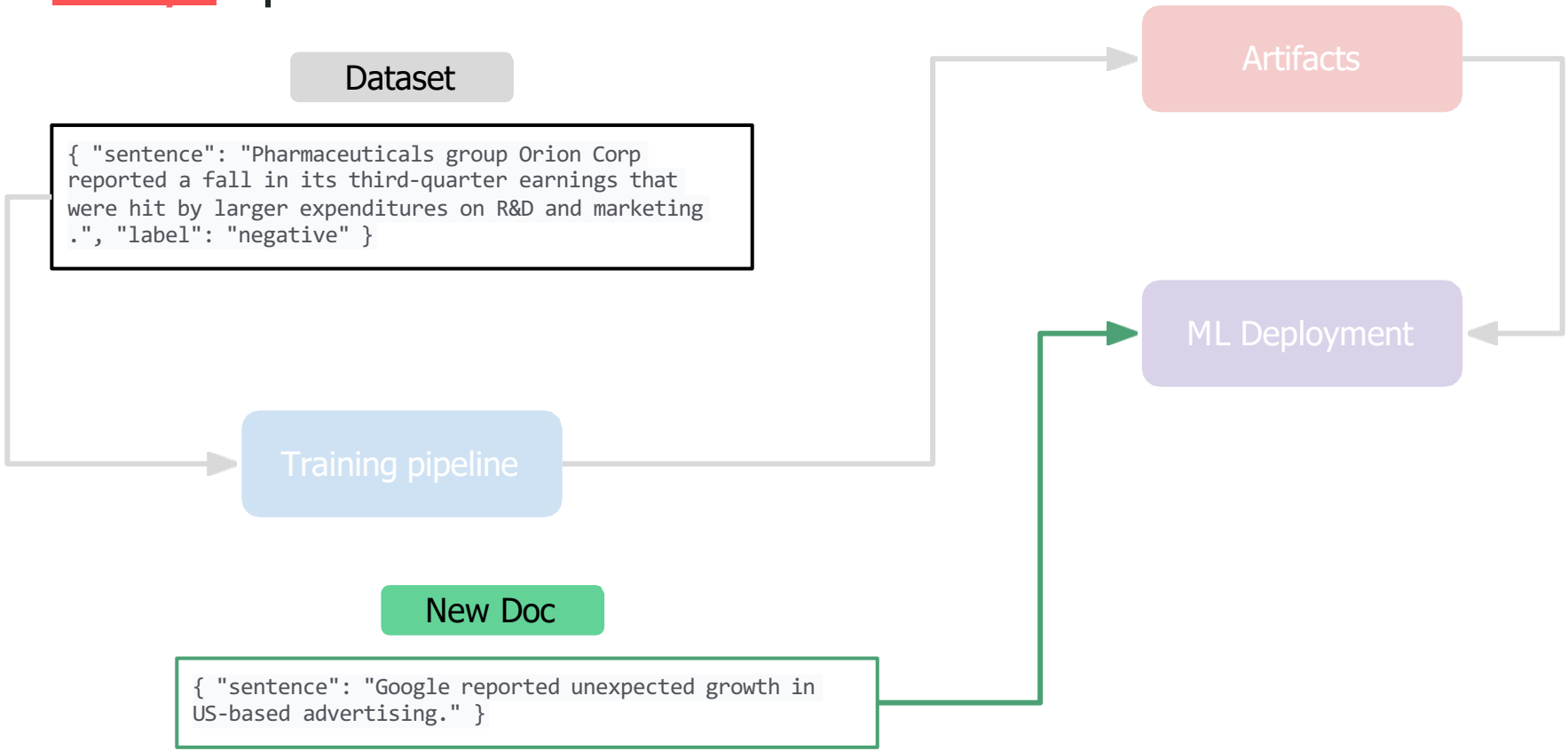
MLSys: үлгісі



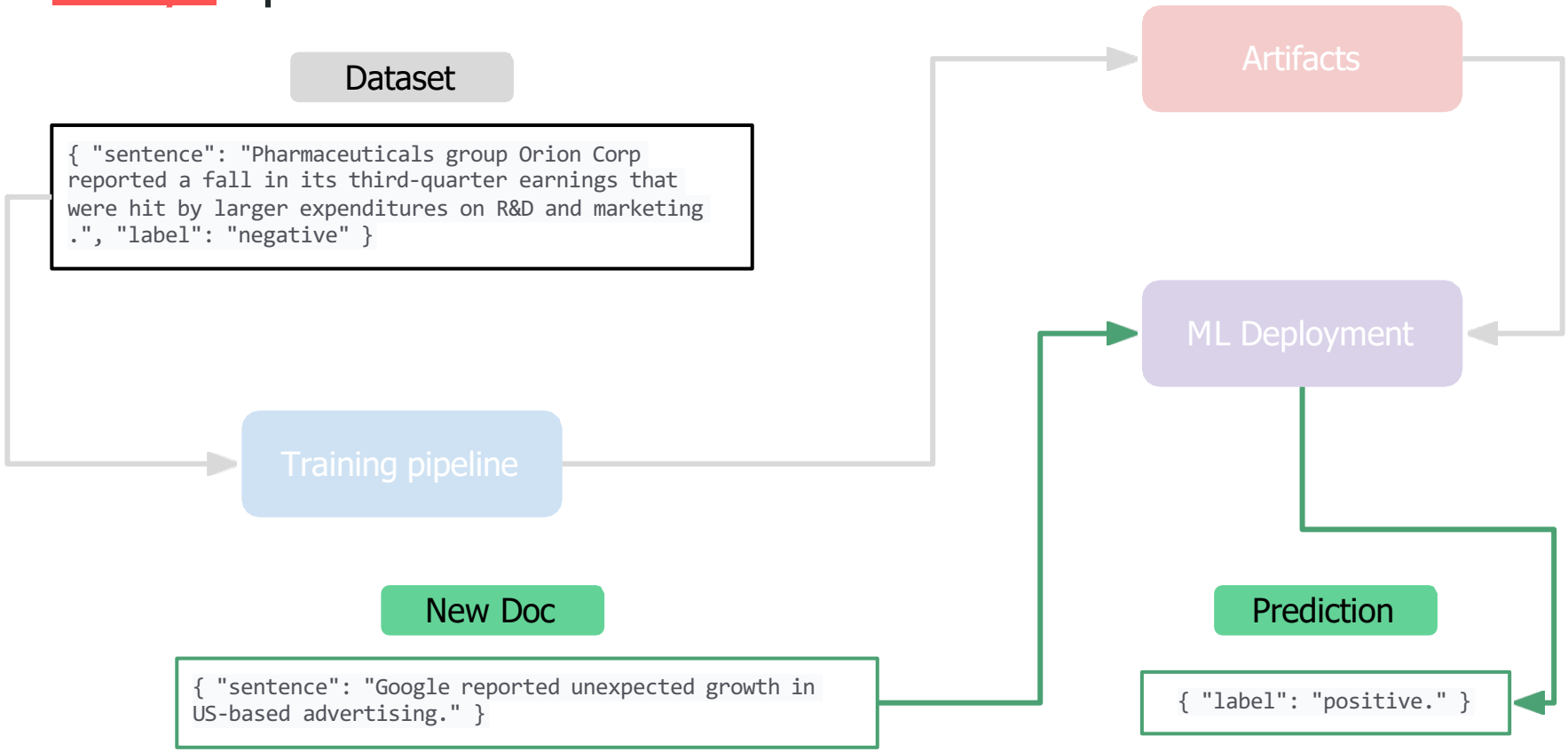
MLSys: үлгісі



MLSys: үлгісі



MLSys: үлгісі



Шынайы әлемдегі ML

Маған шынымен ML керек пе?

Енді біз ML жобаларын талқылайтын болсақ, нақты әлемде сіз ӘРҚАШАН алдымен өзіңізге сұрақ қоюыңыз керек: бұл жоба машиналық оқытуға жарамды ма?

Сіздің жобаңыз ML-ге сәйкес келмеуі мүмкін белгілері мыналарды қамтиды:

- Қарапайым шешімдер трюк жасай алады.
- Деректер жоқ (немесе оны жинаудың практикалық жолы).
- Бір ғана болжау қатесі ауыр салдарға әкелуі мүмкін.
- Жүйенің өнімділігін сенімді түрде өлшеу мүмкін емес.

ML-ді үйрету әдісі нақты әлемдегі сенімділікті қарастырмайды. Біз үйретеміз:

- статикалық деректер жиынын таңдау (MNIST)
- пойызға/сынаққа бөліңіз
- жоғары сынақ шамасына дейін жаттықтырыңыз
- әрі қарай жүр

Әрі қарай жүрудің орнына, біз:

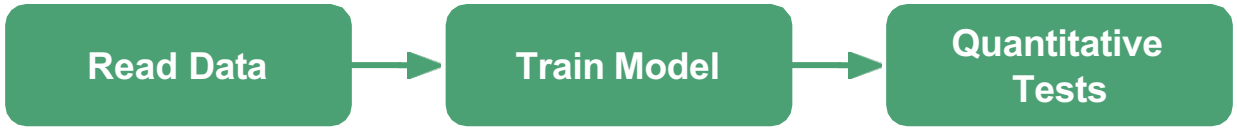
- үлгіні орналастырыңыз
- Пайдаланушыларды үлгілерді бұзуға шақырыңыз
- Мәселелерді шешу үшін деректер жинағын бейімдеңіз

Егер сіздің жұмысыңыз әсер етуі қажет болса, ол **сіздің ноутбугыңыздан тыс жерде жұмыс істеуі керек – нақты әлемдегі жүйелерде, қолданбаларда немесе платформаларда, онда ол нақты нәтижелерге қол жеткізе алады.**

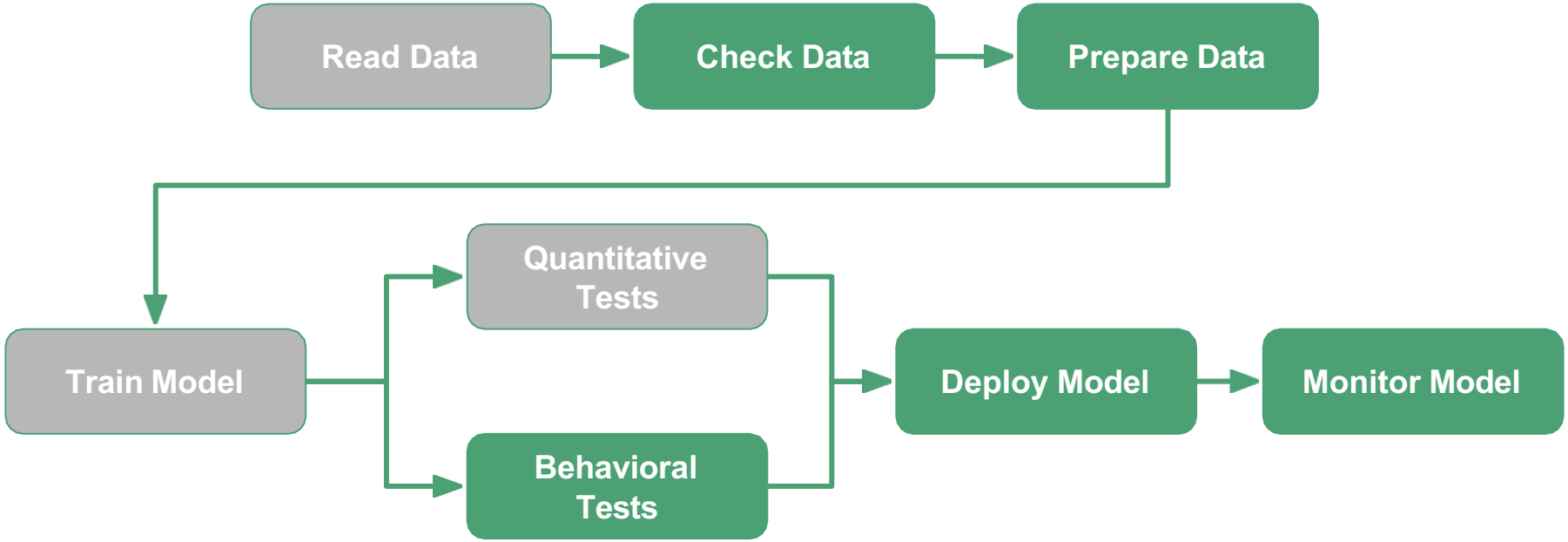
сіздің ноутбугыңыздан тыс жерде жұмыс істеуі керек

1. Сіздің кодыңызды тексеруге, өзгертуге, басқаларға, әдетте техникалық әріптестеріңізге түсінуге болады: таза, модульдік, сыналатын кодты жазуыңыз және құбырды толығымен қайталанатын етуіңіз керек.
2. Сіздің үлгіңізге басқалар, әдетте, техникалық адамдар болуы мүмкін немесе болмауы мүмкін басқа мүдделі тараптар сене алады: оны соңғы пайдаланушылардың алдына итермес бұрын модельдің жобаланғандай әрекет ететініне «тексеру» керек.
3. Болжамдарды басқалар, әдетте интернет қосылымы бар кез келген адам пайдалана алады: үлгіні сәйкес параметрлермен қамтамасыз етілгенде болжамдарды қайтаратын соңғы нүкте ретінде көрсету керек.

Университет vs Шынайы әлем



Университет vs Шынайы әлем



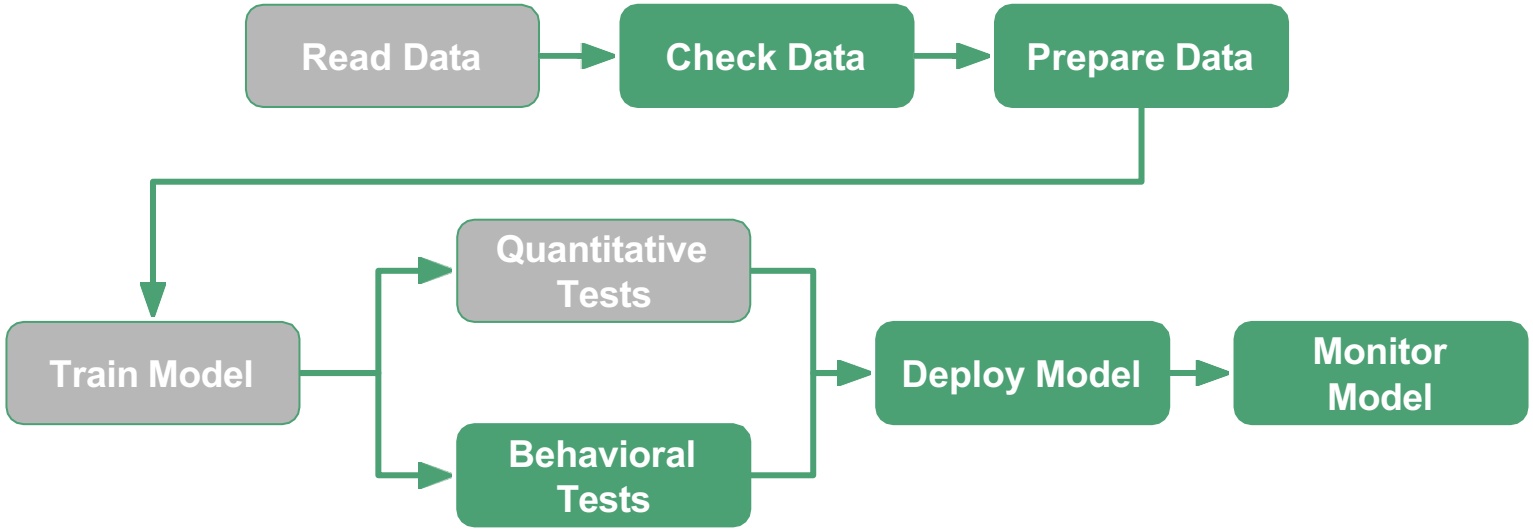
Жобаңызды құрылымдау

Барлығы DAG (бағытталған циклдік график)

- ML жобасы – бұл «қарапайым» қадамдар тізбегі:

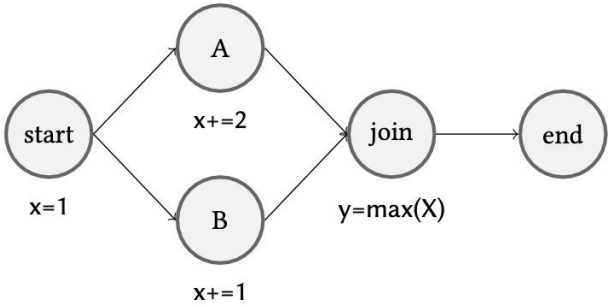
Оның барлық негізгі қадамдары орындалмай тұрып, қадамды орындамау керек;

ЕСКЕРТПЕ: кейбір қадамдар параллельді түрде «тармақталуы» мүмкін (С: ML тілінде оңай параллельдеуге болатын нәрсені ойлай аласыз ба?)



Metaflow-қа жұмсақ кіріспе...

- DAG-дан кодқа және керісінше...



```
class ExampleGraph(FlowSpec):
    @step
    def start:
        self.x = 1
        self.next(self.A, self.B)

    @step
    def A(self):
        self.x += 2
        self.next(self.join)
    @step
    def B(self):
        self.x += 1
        self.next(self.join)
    @step
    def join(self, inputs):
        self.y = max(i.x for i in inputs)
        self.next(self.end)
    @step
    def end(self):
        print("y", self.y)
```

0-бөлім: virtualenv

- ML негізінен Python тілінде жасалады: Интернетте Python тілін үйрену немесе оны жақсырақ меңгеру туралы тамаша оқулықтар/курстар/кітаптар бар. Біз мұнда тек бір негізгі тұжырымдамаға назар аударамыз: виртуалды орта.
- Әртүрлі жобалардың әртүрлі тәуелділіктері болғандықтан, біз оқшаулауды қалауымыз мүмкін
- орталар: ең дұрысы, біз A жобасын тек A, B үшін қажет пакеттермен ғана B және т.б.
- Іс жүзінде бұл нақты жобаларды орындау үшін виртуалды орталарды, таза бөлінген орталарды пайдалану арқылы орындалады: кіріспе үшін calmcode бетін қараңыз.

1-бөлім: кодты құрылымдау

```
def monolith():
    # read the data in and split it
    Xs = []
    Ys = []
    with open('regression_dataset.txt') as f:
        lines = f.readlines()
        for line in lines:
            x, y = line.split('\t')
            Xs.append([float(x)])
            Ys.append(float(y))
    X_train, X_test, y_train, y_test = train_test_split(Xs, Ys, test_size=0.20, random_state=42)
    print(len(X_train), len(X_test))
    # train a regression model
    reg = linear_model.LinearRegression()
    reg.fit(X_train, y_train)
    print("Coefficient {}, intercept {}".format(reg.coef_, reg.intercept_))
    # predict unseeen values and evaluate the model
    y_predicted = reg.predict(X_test)
    fig, ax = plt.subplots()
    ax.scatter(y_predicted, y_test, edgecolors=(0, 0, 1))
    ax.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--', lw=3)
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Actual')
    plt.savefig('monolith_regression_analysis.png', bbox_inches='tight')
    mse = metrics.mean_squared_error(y_test, y_predicted)
    r2 = metrics.r2_score(y_test, y_predicted)
    print('MSE is {}, R2 score is {}'.format(mse, r2))

    # all done
    print("See you, space cowboys!")

    return
```

№1 итерация: монолит

Барлық код бір негізгі скрипте артықшылықтары:

- Тез жазу

Жағымсыз жақтары

- Түсіну қиын (қадамдар арасында логикалық айырмашылық жоқ)
- Ештеңені қайта пайдалануға болмайды
- Тестілеу қиын

1-бөлім: кодты құрылымдау

```python
def composable_script(file_name: str, test_size: float=0.20):
    # all done
    print("Starting up at {}".format(datetime.utcnow()))
    # read the data into a tuple
    dataset = load_data(file_name)
    # check data quality
    is_data_valid = check_dataset(dataset)
    # split the data
    splits = prepare_train_and_test_dataset(dataset, test_size=test_size)
    # train the model
    regression = train_model(splits, is_debug=True)
    # evaluate model
    model_metrics = evaluate_model(regression.model, splits, with_plot=True)
    # all done
    print("All done at {}!\n See you, space cowboys!".format(datetime.utcnow()))

    return


if __name__ == "__main__":
    # TODO: we can move this to read from a command line option, for example
    FILE_NAME = 'regression_dataset.txt'
    TEST_SIZE = 0.20
    composable_script(FILE_NAME, TEST_SIZE)
```

№2 итерация: монолитті бұзу

Тапсырмалар енді бөлек функцияларда

артықшылықтары

- Оқылымдырақ
- Өзгерту, сынау, қайта пайдалану оңай

Жағымсыз жақтары

- Нұсқа жоқ
- Қайта ойнату мүмкіндігі жоқ
- Тапсырманы таңдаулы түрде масштабтау қиын

1-бөлім: кодты құрылымдау

```python
class SampleRegressionFlow(FlowSpec):
    """
    SampleRegressionFlow is a minimal DAG showcasing reading data from a file
    and training a model successfully.
    """

    # if a static file is part of the flow, it can be called in any downstream process, gets versioned etc.
    # https://docs.metaflow.org/metaflow/data#data-in-local-files
    DATA_FILE = IncludeFile(
        'dataset',
        help='Text file with the dataset',
        is_text=True,
        default='regression_dataset.txt')

    TEST_SPLIT = Parameter(
        name='test_split',
        help='Determining the split of the dataset for testing',
        default=0.20
    )

    @step
    def start(self):
        """
        Start up and print out some info to make sure everything is ok metaflow-side
        """
        print("Starting up at {}".format(datetime.utcnow()))
        # debug printing - this is from https://docs.metaflow.org/metaflow/tagging
        # to show how information about the current run can be accessed programmatically
        print("flow name: %s" % current.flow_name)
        print("run id: %s" % current.run_id)
        print("username: %s" % current.username)
        self.next(self.load_data)
```

№3 итерация: Метаағыны Тапсырмалар енді DAG

Артықшылықтары

- Толығымен модульдік
- Әр тапсырма бойынша іріктеп масштабтау
- Барлығы нұсқада және қайта ойнатылады

Жағымсыз жақтары

- Қосымша күрделілік

Метаағын ортақ лексикон ретінде

1. **Ағын**: құбырдың өзін сипаттайтын DAG.
2. **Іске қосу**: DAG орындалған сайын ол жаңа іске қосылады. Жүгірулер оқшауланған және аттар кеңістігі, мысалы. user:jacopo vs user:ethan ретінде тегтелген іске қосулар бірдей ағын болуы мүмкін, бірақ әртүрлі адамдармен орындалады.
3. **Қадам**: DAG түйіні.
4. **Тапсырма**: оқшауланған және дербес қадамды орындау.
5. **Артефакт**: іске қосу арқылы жасалған және метадеректер қоймасында нұсқаланған кез келген деректер/үлгі/күй (мысалы, myFlow/12/training/dataset).
6. **Client API**: Python негізіндегі интерактивті режим, онда отладтау және визуализация мақсаттары үшін барлық іске қосулардың метадеректері мен артефактілерін тексеруге болады.

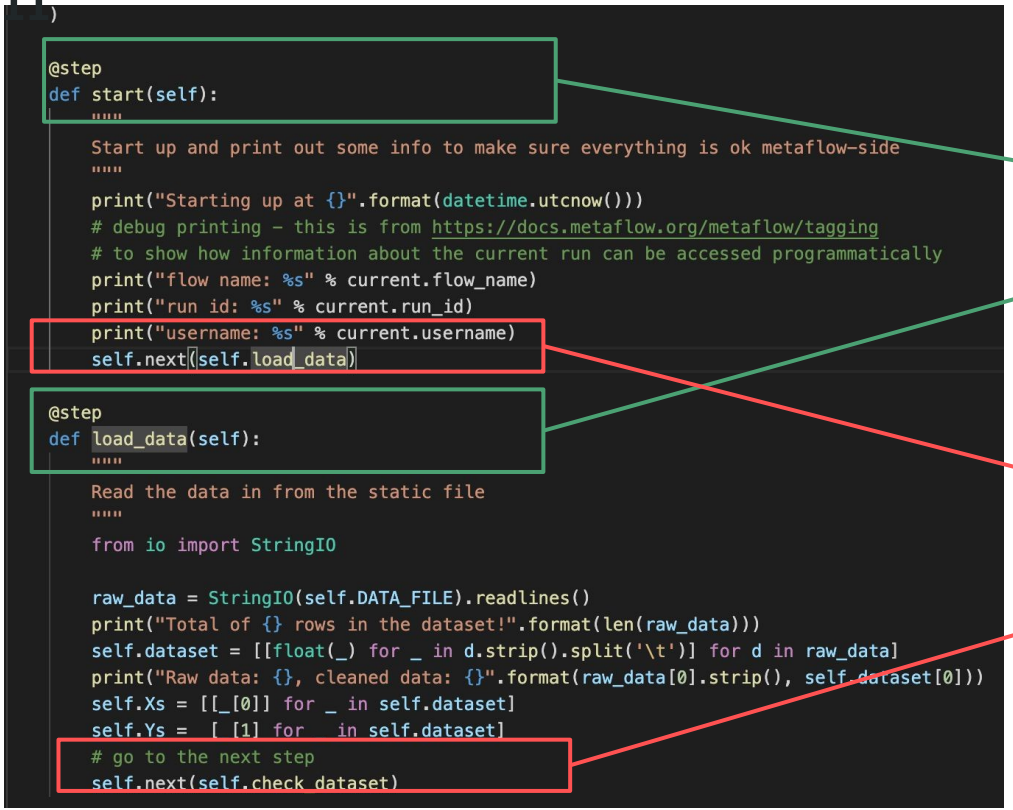
Metaflow жобалары (арнайы) Python сыныптары ретінде - I

```python
class SampleRegressionFlow(FlowSpec):
    """
    SampleRegressionFlow is a minimal DAG showcasing reading data from a file
    and training a model successfully.
    """

    # if a static file is part of the flow,
    # it can be called in any downstream process,
    # gets versioned etc.
    # https://docs.metaflow.org/metaflow/data#data-in-local-files
    DATA_FILE = IncludeFile(
        'dataset',
        help='Text file with the dataset',
        is_text=True,
        default='regression_dataset.txt')

    TEST_SPLIT = Parameter(
        name='test_split',
        help='Determining the split of the dataset for testing',
        default=0.20
    )
```

A project class inheriting from FlowSpec

OPTIONAL: Parameters to configure the flow, Files as input

Metaflow жобалары (арнайы) Python сыныптары ретінде - II

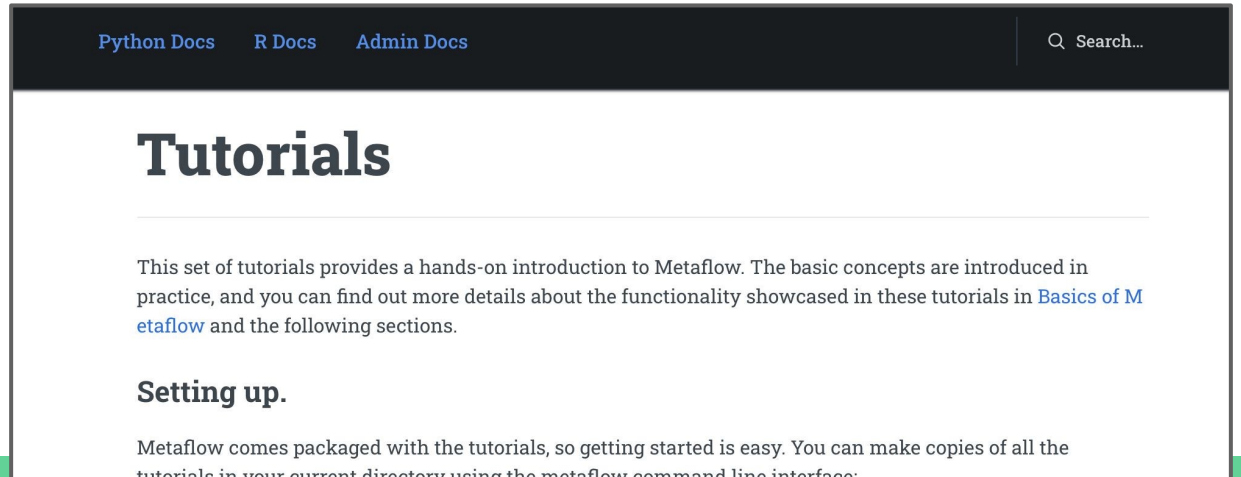


Functions decorated with @steps: each function is a node in the DAG

Each function lists its descendant(s) through the next command.

Метаағын(metaflow) компоненттері

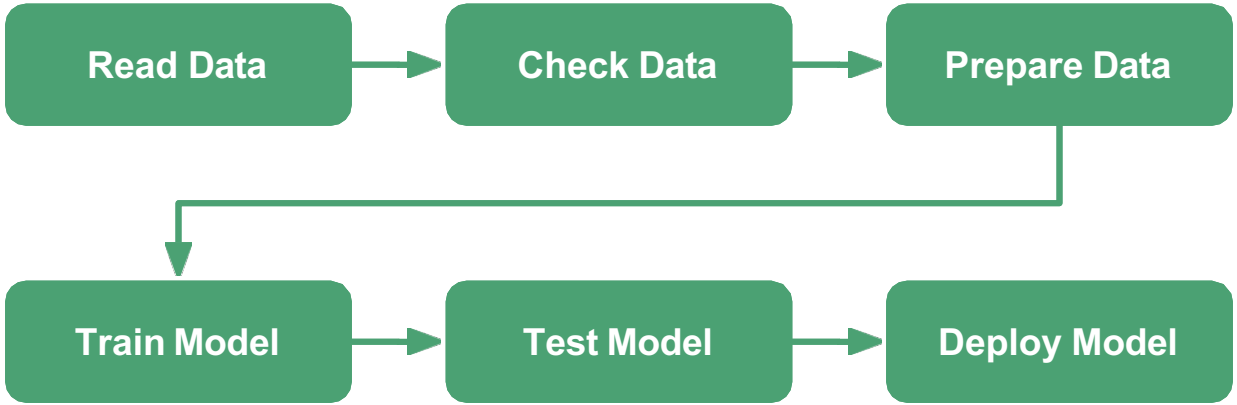
1. **DAG анықтамасы:** біз не істеп жатырмыз? Қадамдар, тәуелділіктер, параллелизация және т.б.
2. **Metastore:** заттарды қайда сақтаймыз? Айнымалылар, күйлер, метадеректер және т.б.
3. **Есептеу қабаты:** есептеуді не орындайды? Ресурстар, бұлттық құралдар және т.б.



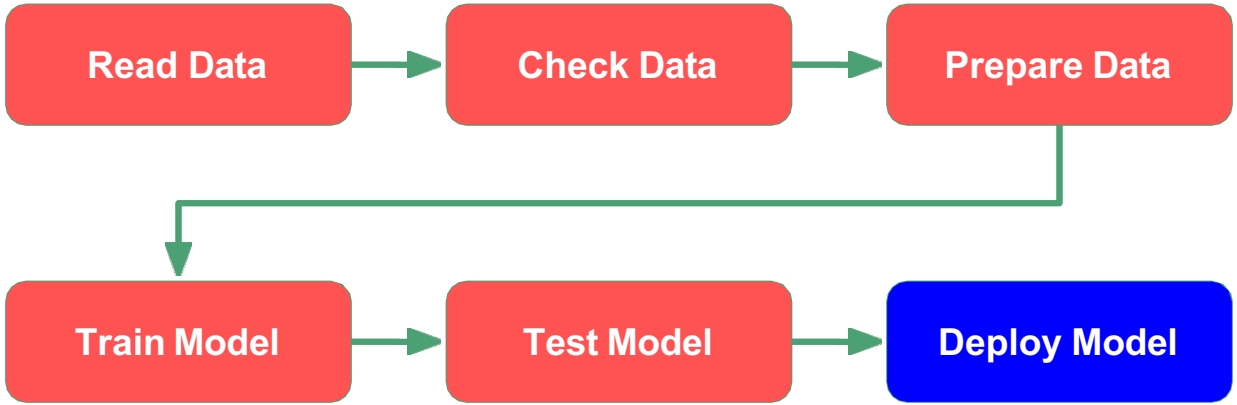
4 принциптегі метаағым

№1: ML жобалары DAG болып табылады

Тапсырмалар басқа тапсырмалардың ішкі жиынына ғана байланысты: параллелизация мүмкін және сәтсіздікке ұшыраған жағдайда әрекетті қайталау ақылды болуы мүмкін!



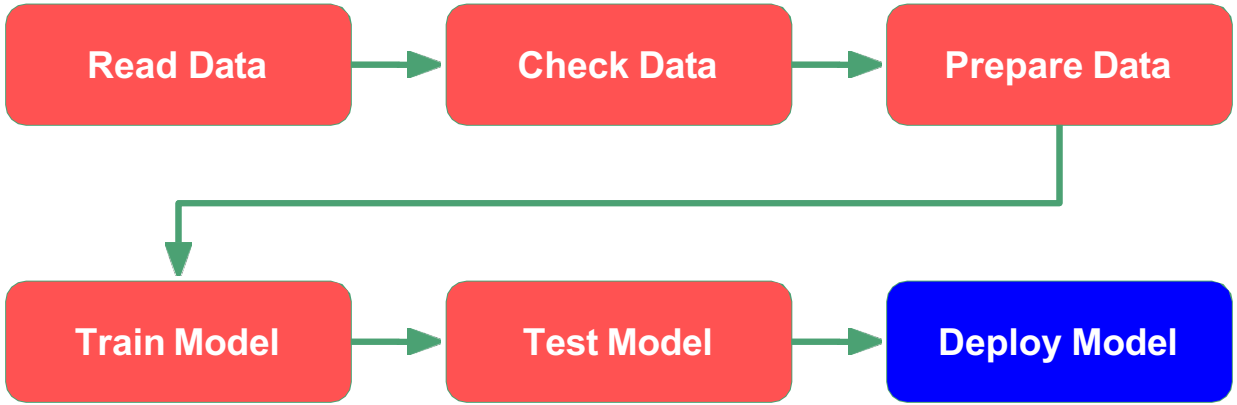
Біз ML жобамыздың екі фазасын ажыратамыз: оқыту кезеңі (жүктеме деректері, деректерді тексеру, оқыту және тестілеу үлгісі...) және қызмет көрсету кезеңі (үлгі болжамын басқа пайдаланушыларға көрсету).



Бұл сыныпта (сонымен қатар саладағы жаңа жобаларды әзірлеу кезінде) бізде:

оқыту кезеңі: жергілікті орындалады (Metaflow ішінде)

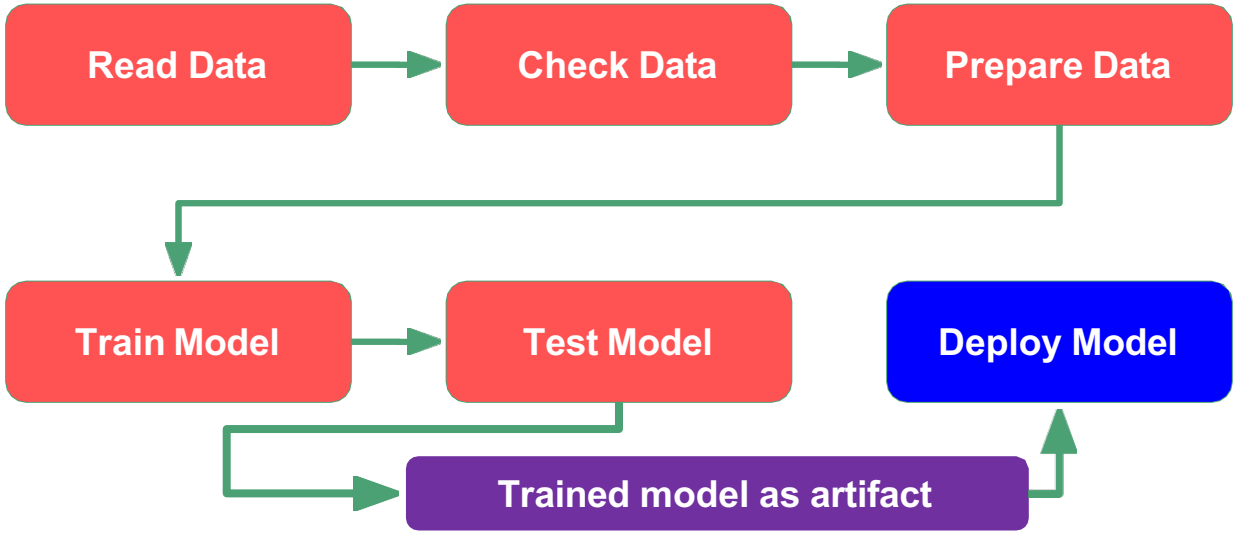
қызмет көрсету кезеңі: бұлтта орындалды (AWS-де)



Бұл сыныпта (сонымен қатар саладағы жаңа жобаларды әзірлеу кезінде) бізде:

оқыту кезеңі: жергілікті орындалады (Metaflow бағдарламасында) және үлгі артефакт жасайды

қызмет көрсету кезеңі: бұлтта орындалды (AWS-де)



4 принциптегі метаағым

№2: Деректер мен күйлер ML құбырларының бөлігі болып табылады (нұсқалау, қайта ойнату)

```
@step
def load_data(self):
    """
    Read the data in from the static file
    """
    from io import StringIO

    raw_data = StringIO(self.DATA_FILE).readlines()
    print("Total of {} rows in the dataset!".format(len(raw_data)))
    self.dataset = [[float(_) for _ in d.strip().split('\t')] for d in raw_data]
    print("Raw data: {}, cleaned data: {}".format(raw_data[0].strip(), self.dataset[0]))
    self.Xs = [[_[0]] for _ in self.dataset]
    self.Ys =  [_[1] for _ in self.dataset]
    # go to the next step
    self.next(self.check_dataset)
```

The raw dataset is saved!

The X,Y dataset is saved!

4 принциптегі метаағым

№2: Деректер мен күйлерді әрқашан тексеруге болады (дәптерді тексеріңіз)

Get artifacts from latest successful run

In [4]:
```
def get_latest_successful_run(flow_name: str):
    "Gets the latest successfull run."
    for r in Flow(flow_name).runs():
        if r.successful:
            return r
```

In [5]:
```
latest_run = get_latest_successful_run(FLOW_NAME)
latest_model = latest_run.data.model
latest_dataset = latest_run.data.dataset
```

Verify we can inspect the dataset...

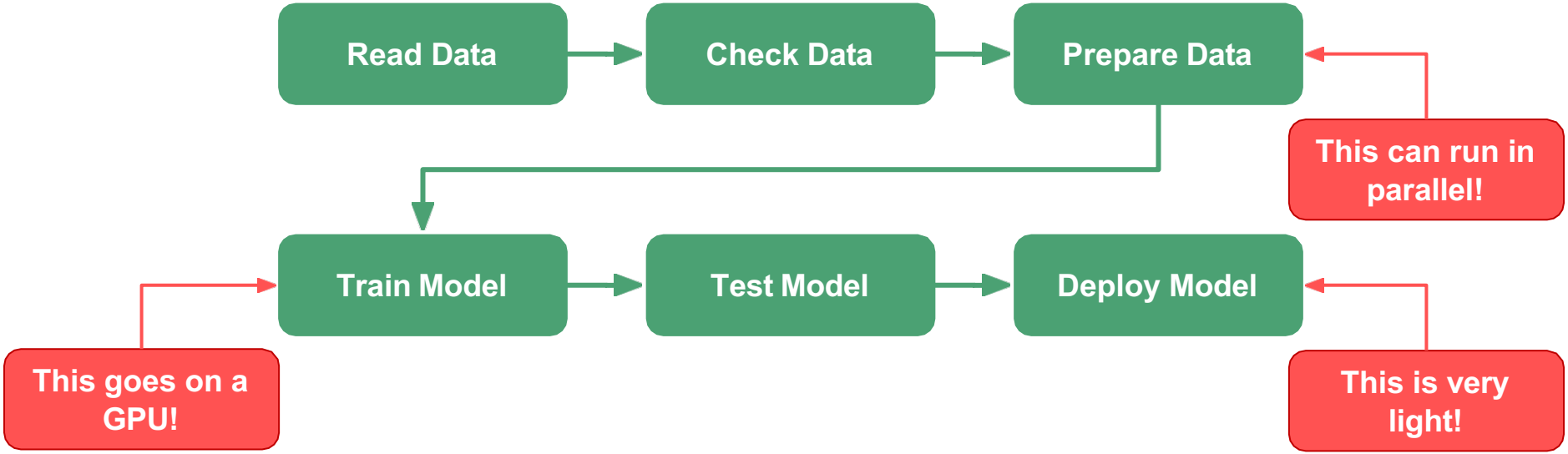
In [6]:
```
latest_dataset[:10]
```

Out[6]:
```
[[-1.7587394864231143, -32.770386047959725],
 [1.0318445394686349, 3.5045910648442344],
 [-0.48760622407249354, -17.930307666159294],
 [0.18645431476942764, -3.990201236512462],
 [0.725766623898692, 13.105264342363048],
 [0.9725544496267299, 33.7844061138283],
 [0.6453759495851475, -6.568374494070948],
```

4 принциптегі метаағым

№3: Бір есептеу өлшемі барлығына сәйкес келмейді

Қажет болғанда ғана жергілікті және бұлттық есептеулер арасында ауыса отырып, әр тапсырма үшін есептеу ресурстарын (және пакеттерді) анықтауға болады.



4 принциптегі метаағым

№4: Сіз команданың мүшесі болсаңыз, бәрі керемет

Бірнеше пайдаланушылар бір ағынды бірге іске қоса алады, содан кейін топ барлық іске қосулар арқылы тәуелсіз жасалған артефактілерді талдай алады.